МОНИТОР

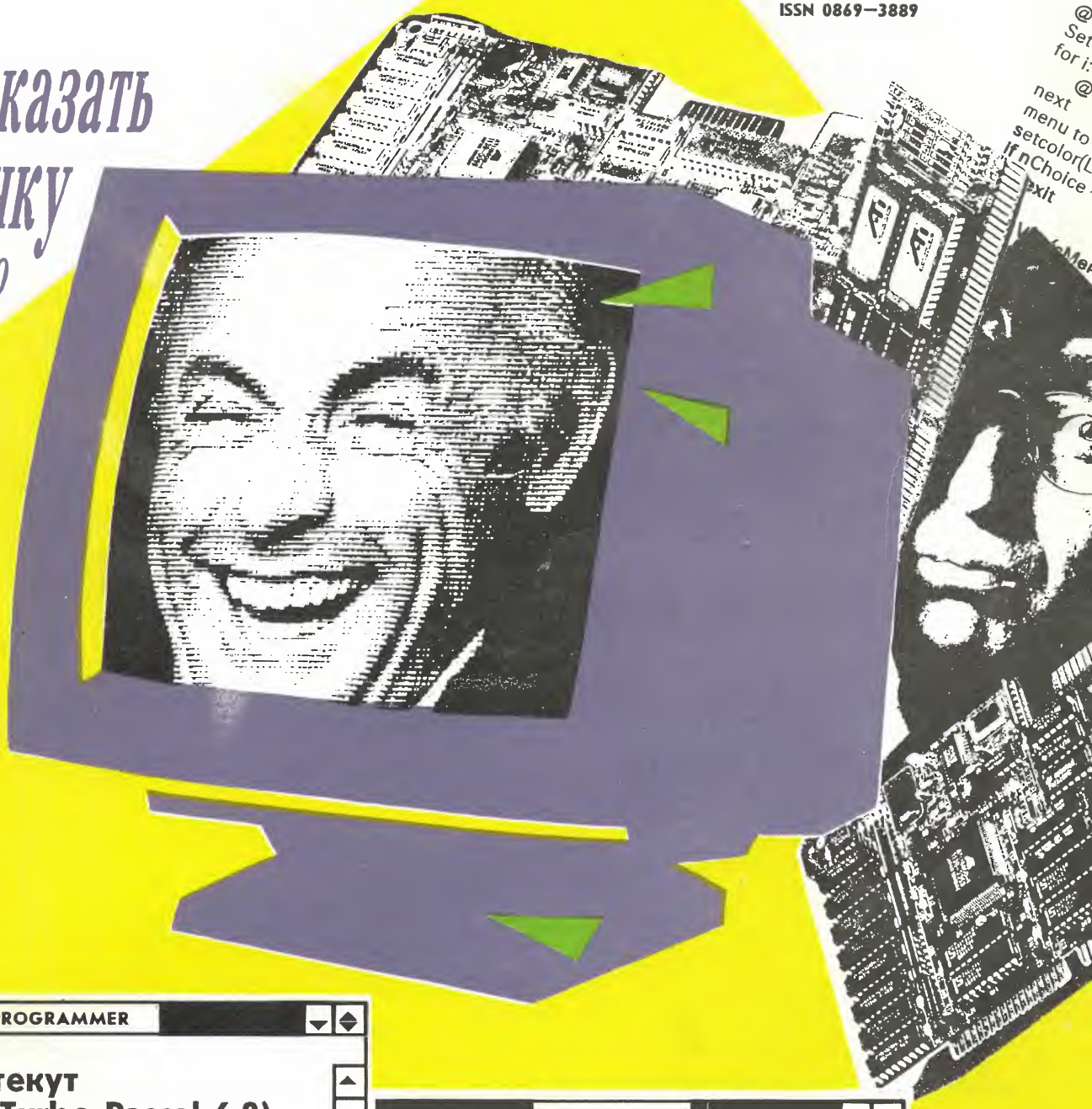
журнал для программистов — практиков

3.92

ISSN 0869—3889

Показать картинку

стр. 20



PROGRAMMER

Все реки текут (потоки в Turbo Pascal 6.0)

Препроцессор Clipper 5.0

SALT— язык телекоммуникаций

USER

AutoCAD 11— что нового!

Ты записался в убийцы Джафара!!!

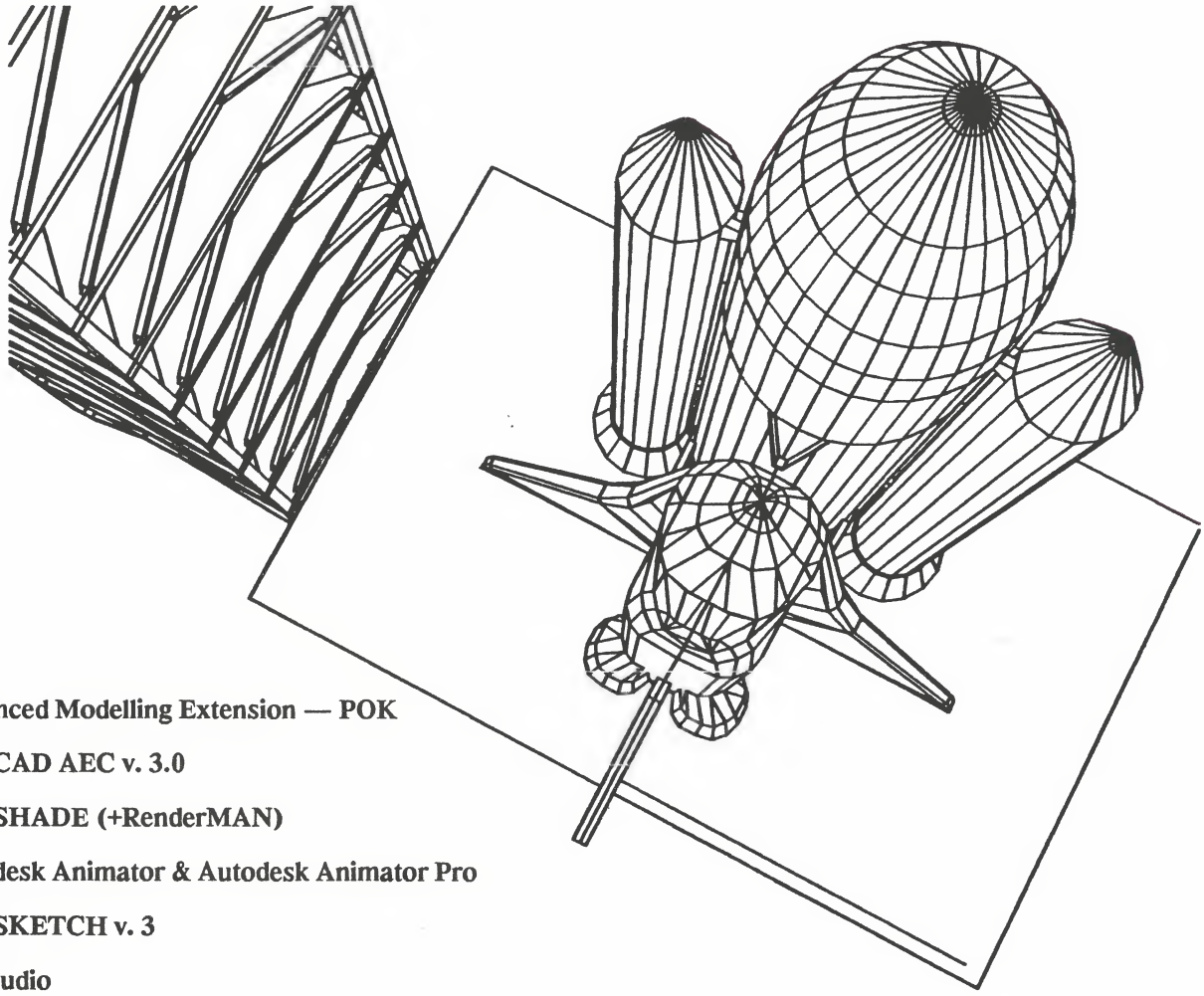
Autodesk RF

предлагает Вашему вниманию следующие программные продукты

Лучшая в мире система автоматизации проектно-конструкторских работ

Автокад версии 11

На русском или английском языке, за рубли или за СКВ



Advanced Modelling Extension — POK

AutoCAD AEC v. 3.0

AutoSHADE (+RenderMAN)

Autodesk Animator & Autodesk Animator Pro

AutoSKETCH v. 3

3D Studio

Generic CADD 5.0 Plus

Generic 3D Drafting

James Gleick's CHAOS: The Software

CA Lab

Российская федерация
107005, Москва,
2-я Бауманская ул., 9/23
Телефон: (095) 261 63 63
Факс: (095) 267 99 27

Компания Autodesk RF имеет развитую сеть дилеров, официальных учебных центров и авторизованных разработчиков

Приглашаем к сотрудничеству!

"очень своевременная книга!"

ОБШИРНЫЙ **КАТАЛОГ** ЗАРУБЕЖНЫХ ФИРМ

Электроника для Офисов, Банков и Торговли
Бытовая Электроника, Компьютеры и Периферия
Разнообразное Программное Обеспечение

Адреса, Телефоны
Предлагаемые Товары

Стоимость каталога - 985 рублей + *28%*
Для тех, кто оплатит заказ до 20 июля - скидка 15 %, 2 экземпляра - скидка 20 %

Бланки заказов и копии платежных поручений просьба высылать по адресу:

Москва, 117419, а/я 765, Каталог

Счет: НТО "Софт-Москва" р/с 467002 в КБ "Банк Развития XXI век", к/с 161816 в РКЦ ГУ ЦБ РФ по г.Москва, МФО 201791

Рассылка каталога начинается в III квартале *тел. (095) 247 3625*

БЛАНК ЗАКАЗА

Название организации _____

Адрес для высылки Каталога _____

_____

Телефон _____ Количество Экземпляров _____

МОНИТОР

Журнал для программистов-практиков

Редколлегия:
Д.Э.Бройтман (главный редактор)
Р.А.Болдырев
В.В.Водолазский
М.И.Гетманская (отдел рекламы)
И.Г.Гун
С.В.Макельский (главный художник)
А.В.Орехов (издательский директор)
А.В.Свириденко
А.В.Семенников
Л.И.Устинова
А.А.Якубенко (ответственный секретарь)

В оформлении номера использованы рисунки
А.С.Замулы

Подписано в печать с оригинал-макета 26.05.92.
Формат 60x90/8.
Тираж 30 000 экз.
Печать офсетная. Печ.л.8.0.
Изд. N1971.
Цена договорная

Издатель:
Научно-техническое общество

Софт·Москва

Для писем: 117419, г.Москва,а/я 765,
тел. (095) 237-21-36, 247-36-25,
факс 126-31-56

© "Монитор", 1992

Операционные системы

Ю.Крючков. **Опыт работы с оболочкой ViewMax операционной системы DR DOS 5.0** 4

Описаны основные функции файловой оболочки ViewMax. Приведены результаты сравнительного анализа с популярными программами Norton Commander и Pie Commander.

Языки программирования

А.Брюзгин. **Чемпионат по Windows - программированию. Победил Digitalk - выиграли все** 9

В статье рассказывается о соревнованиях по Windows-программированию, в которых приняли участие 6 фирм, производящих высокоуровневые средства создания Windows-программ.

Базы данных

А.Чернышков. **Clipper 5.0 - новые возможности Часть 3. Препроцессор** 15

Описывается препроцессор, используемый в Clipper 5.0 для предварительной обработки текстов программ перед компиляцией.

Мир графики

А.Тихонов. **Показать картинку...** 20

В статье рассмотрены графический формат PCX и средства для работы с ним в Turbo Pascal версии 6.0. Приводятся листинги программ, позволяющие демонстрировать образы, созданные в графическом редакторе PC PaintBrush и хранящиеся в файлах, библиотеках и оперативной памяти.

Технологии программирования

Д.Арапов. **Объектно-ориентированный ввод-вывод в Turbo Pascal 6.0. Последовательный доступ** 30

В статье изложена концепция потоков в Turbo Pascal 6.0. Описывается организация последовательного доступа к памяти.

Лицей

С.Курковский. **Меню как основа технологии взаимодействия с пользователем** 36

Рассматриваются вопросы технологии программирования меню как основы для удобного диалога прикладных программ с пользователем. Разработана универсальная система поддержки меню, программно реализованная на языке Турбо Си.

САПР

С.Париевский. **Система автоматизированного проектирования Автокад версии 11** 43

Первая статья из серии статей, посвященных программированию в среде Автокад.

Защита программ — В.Вегнер, А.Крутяков, В.Серегин, В.Сидоров, А.Спесивцев — **Анализ отечественных систем защиты программ от копирования. Часть 2. Война форматов** — **48**

Анализ форматов ключевых дискет ряда отечественных систем защиты от копирования.

Дайджест — **Зарубежная компьютерная хроника** — **51**

Терроризм — В.Водолазкий. — **Индикатор заражения программы компьютерным вирусом** — **52**

Статья содержит описание программы, позволяющей обнаружить факт вирусной атаки по изменению длины исполняемой программы. Языки программирования - Turbo Pascal 5.0+ и Turbo Assembler 2.0+.

Коммуникации — Е.Равич. — **Выходите на связь. Часть 3. Пишем сценарии** — **54**

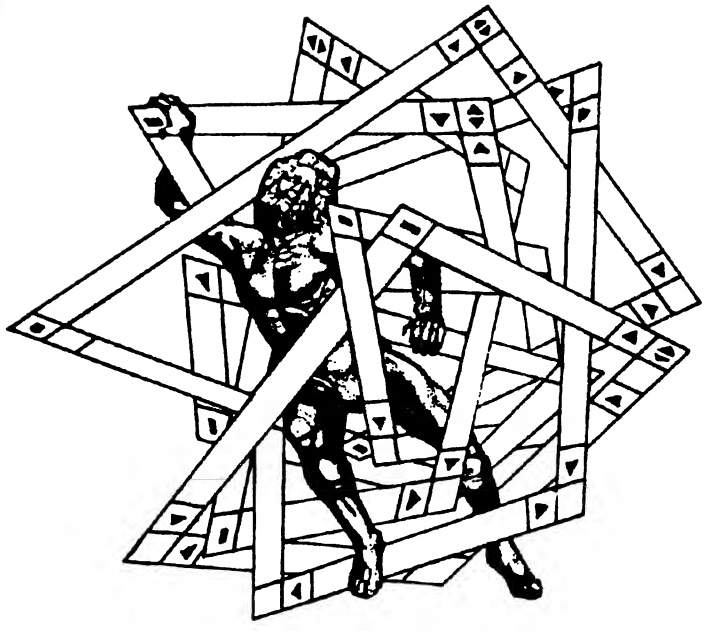
Статья рассказывает о языках для написания коммуникационных сценариев SALT (программа Telix) и TMScript (программа Telemate). Приводится листинг сценария, написанного на языке SALT.

Текстовые редакторы — А.Дайховский. — **Новое лицо редактора ChiWriter - версия 4!** — **61**

Впечатления пользователя от работы с новой версией текстового редактора ChiWriter.

Игры — А.Лукашук. — **"Ты записался в убийцы ДЖАФАРА ???" Полезные советы любителям игры "Prince of Persia"** — ***64***

Описаны возможности, предоставляемые игроку "Prince of Persia" при использовании программного люка, оставленного разработчиками игры.



Опыт работы с оболочкой ViewMAX операционной системы DR DOS 5.0

Ю.Крючков

Во ViewMAX реализован достаточно интересный метод представления файлов и каталогов

Почти все пользователи персональных компьютеров применяют при работе оболочки DOS. Оболочки DOS обеспечивают пользователю больший уровень комфортности по сравнению с выполнением аналогичных действий с использованием команд операционной системы. Существует достаточно большое число оболочек DOS. Среди наиболее известных - Norton Commander и ее разновидность Pie Commander. При сравнении оболочек обычно рассматривают следующие характеристики: на какую версию DOS и какие типы компьютеров они ориентированы, какой объем оперативной памяти необходим для работы оболочки, какой объем резидентного модуля остается в памяти при запуске из оболочки прикладной программы, какие функции можно выполнить в оболочке при работе с файлами и каталогами, какие файлы можно просмотреть из оболочки, как можно запускать прикладные программы из оболочки, используя систему меню, а также другие подобные действия и функции. И если наиболее популярным оболочкам посвящено уже немало публикаций, то оболочка ViewMAX практически неизвестна для большинства пользователей.

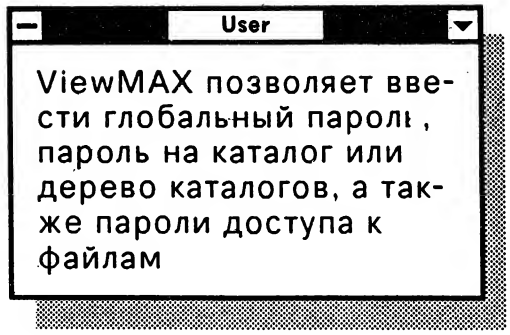
Описываемая здесь оболочка DOS ViewMAX входит в поставку операционной системы DR DOS версии 5.0. Программа начальной установки операционной системы DR DOS 5.0 позволяет установить и настроить оболочку ViewMAX под имеющиеся типы монитора и манипулятора "мышь". Это достаточно важный этап настройки, ибо если ошибиться в типе адаптера или драйвера манипулятора, то оболочка будет функционировать неверно, а исправить ошибку можно лишь повторив всю операцию настройки оболочки ViewMAX и установки операционной системы DR DOS 5.0.

ViewMAX - программный продукт фирмы Digital Research. Оболочка ориентирована на компьютеры IBM PC, PS/2 или совместимые с ними персональные компьютеры, версию дисковой операционной системы DR DOS 5.0 (желательно наличие жесткого диска и не менее 384 Кбайтов оперативной памяти). При запуске из оболочки ViewMAX прикладной программы в оперативной памяти остается резидентный модуль объемом 14,7 Кбайтов. Оболочка может быть использована для работы с сетевыми системами (например Novell).

Оболочка ViewMAX работает в графическом режиме. На установочной дискете есть драйверы для видеоадаптеров типа Hercules, CGA, EGA-VGA. При наличии манипулятора "мышь" оболочка самостоятельно без загрузки драйвера поддерживает манипуляторы Microsoft Serial и Logitech Mouse (или совместимые с ними), а также манипуляторы IBM

PS/2 Mouse и Bus Mouse при загрузке драйвера MOUSE.SYS. Например, трехкнопочная "мышь" Genius GM-6 прекрасно совместима с оболочкой ViewMAX, если ее объявить при настройке как Logitech Mouse.

После установки оболочки на диске должны остаться 9 файлов объемом не более 250 Кбайтов. К ним относятся драйвер экрана, экранные шрифты, оверлейные модули, модули встроенного калькулятора, часов-"будильника" и некоторые другие.



Во ViewMAX реализован достаточно интересный метод представления файлов и каталогов. Кроме традиционных режимов Text (файлы и каталоги представляются в текстовой форме с полным именем и расширением, датой создания, объемом - для файлов, атрибутами) и Tree (логический диск или часть диска выводится в виде дерева каталогов) есть режим Icon, когда файлы и каталоги представлены в виде пиктограмм с именем и расширением. Имеется набор пиктограмм для обозначения файлов баз данных, электронных таблиц, графических систем, проектов и т.п. Представленные на экране пиктограммы говорят нам о функциональном назначении того или иного файла (рис.1).

Экран при работе с оболочкой ViewMAX разбит на следующие рабочие зоны: сверху строка основного меню, при выборе одного из пунктов которого появляется меню более низкого уровня; ниже область или две области отображения содержимого дисков или дискового пространства. Область отображения содержит в верхней части название, полный путь, объем и шаблон для файлов, кнопки для возврата на более высокий уровень и для распахивания/ сжатия области на весь или пол-экрана; в правой части - линейку листания с соответствующими кнопками и основное поле отображения. Вид экрана при работе с оболочкой ViewMAX представлен на рис.2.

ViewMAX представляет два уровня отображения: логические диски, присутствующие в системе (выход в сеть – отдельный логический диск), и содержимое логического диска (каталоги, подкаталоги, файлы).

Операции с файлами и каталогами

Оболочка ViewMAX реализует основные функции управления файлами и каталогами, а также представляет некоторые новые возможности, присущие операционным системам DR DOS. При помощи оболочки можно выбирать файлы и подкаталоги текущего каталога для последующего их удаления или копирования в пределах дискового пространства или между дисками. Подкаталоги, отмеченные для копирования или удаления, могут содержать целые ветви дерева каталогов.

Можно изменить имя файла или каталога, установить или снять пароль, обеспечивающий три уровня доступа к файлу, запретить паролем доступ к каталогу или дереву каталогов. Подобно утилите PASSWORD операционной системы DR DOS, ViewMAX позволяет самостоятельно ввести глобальный пароль, пароль на каталог или дерево каталогов, а также три уровня доступа к файлам: 1-й – невозможность чтения, копирования, изменения, удаления и переименования файла или группы файлов; 2-й – невозможность копирования, стирания и переименования; 3-й – невозможность удаления или переименования (вы не можете в этом случае модифицировать файл или группу файлов). И наконец, можно установить комбинированый пароль, когда будут защищены не только файлы внутри каталога, но и сам каталог, и введен глобальный пароль.

В тоже время оболочка ViewMAX не позволяет осуществлять поиск файлов или наборов данных, сравнивать каталоги между собой и отмечать точки несовпадения. Кроме этого к недостаткам оболочки можно отнести тот факт, что нельзя отметить все файлы каталога, если они не помещаются на экране в области отображения. При отображении на экране двух дисков в текстовом режиме в области отображения помещаются 11 файлов, в пиктограммном - 28. При отображении одного диска в распахнутом окне - соответственно 25 и 63 файла. Но наиболее существенной недоработкой ViewMAX является невозможность переноса файлов и каталогов.

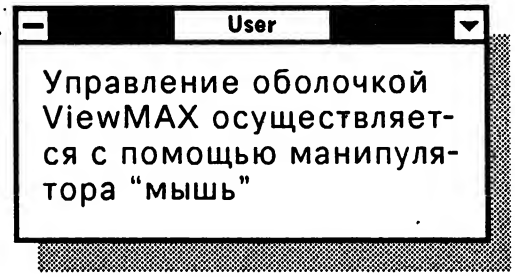
Просмотр содержимого файлов

Оболочка ViewMAX не предоставляет больших возможностей для просмотра содержимого файлов. Она обеспечивает просмотр в коде ASCII и шестнадцатиричном. При этом не обеспечивается возможность свертки текста и для просмотра всей информации нужно прокручивать текст по горизонтали. Кроме того, так как вывод данных на экран происходит в графическом режиме при помощи специального драйвера и файла шрифтов, а в нем отсутствуют знаки кириллицы, русский текст будет отображаться неверно - знаки кириллицы заменяются знаками второй половины кодовой таблицы.

Запуск прикладных программ пользователя

Запуск прикладных программ пользователя из оболочки непосредственно связан с конфигурированием пиктограмм. Как известно, файлы могут быть исполняемыми (программы) и неисполняемыми (данные). Конфигурирование пиктограмм заключается в связи головного исполняемого файла прикладной программы с файлами набора данных, относящихся к этой программе (см. рис.1). Например, если Вы используете текстовый редактор ЛЕКСИКОН, то применяя режим сболочки Options/Program application, можно присвоить файлу lex.exe пиктограмму текстового редактора и связать с ним файлы, имеющие расширение lex, txt и doc с при-

своением им соответствующей оригинальной пиктограммы. В дальнейшем после указания пути доступа к головному файлу редактора в autoexec.bat нажатие на <Enter> или левую клавишу "мыши", когда курсор расположен на любом файле, имеющем расширение lex, txt или doc, вызовет запуск прикладной программы - ЛЕКСИКОН с загрузкой этого файла в текстовый редактор. Оригинальные пиктограммы в оболочке ViewMAX имеют логические диски, сетевой диск, каталоги, исполняемые и командные файлы, а также файлы наборов данных. Для исполняемых файлов и файлов наборов данных - текстовых и графических редакторов, систем управления базами данных, рабочих (инструментальных) программ, интегрированных и издательских систем, связи, электронных таблиц и т.п. в оболочке имеются особые пиктограммы.



К дополнительным возможностям оболочки ViewMAX можно отнести то, что в ней можно осуществить копирование всего логического диска; не выходя из оболочки, выполнить форматирование дискет; просмотреть объем логического диска с одновременным показом числа каталогов и файлов на нем, а также занятого и свободного пространства на диске; сменить имя файла или каталога. В оболочку введены калькулятор для выполнения простейших математических вычислений и часы с функцией "будильник". Причем местоположение часов и калькулятора на экране можно легко изменить, захватив название манипулятором "мышь" и "перетащив" на новое место. К сожалению, оболочка не имеет встроенного текстового редактора, а для задания команды из командной строки нужно выполнить последовательность операций Options/Enter DR DOS command/ и после запуска новой версии командного процессора набрать команду. После выполнения программы возврат в оболочку происходит через EXIT.

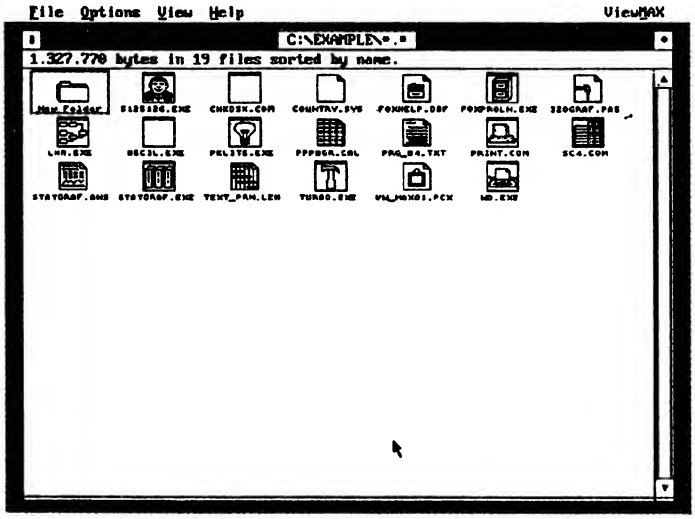


Рис.1. Примеры пиктограмм, используемых в оболочке ViewMax

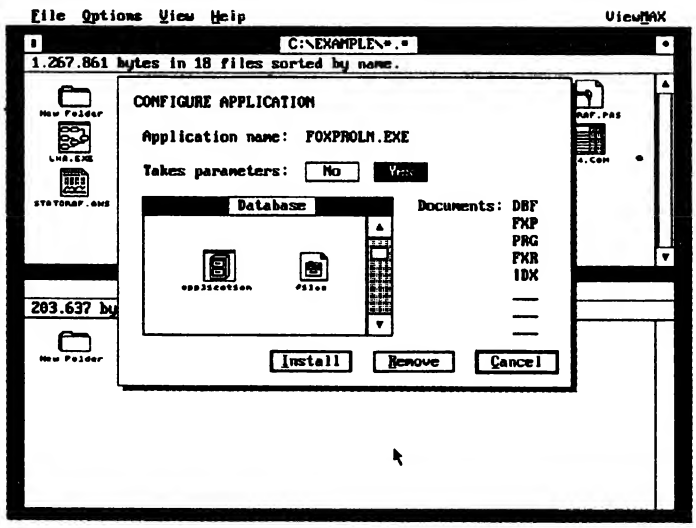


Рис.2. Вид экрана при работе с оболочкой ViewMax

Управление оболочкой

Управление оболочкой ViewMAX достаточно простое и осуществляется нажатием левой клавиши манипулятора "мышь" при установке стрелки на выбранный пункт меню (линейку листания или кнопку). Раскрытие самих меню осуществляется автоматически при подводе стрелки к основному меню. При отсутствии манипулятора раскрыть меню или выполнить отмеченный пункт меню можно нажатием клавиши <Alt+подчеркнутая буква в названии пункта меню>. Отметить файл, каталог или диск можно клавишей <Пробел>, при этом высвечивается соответствующая пиктограмма или инвертируется имя в режиме показа списка файлов или дерева каталогов. Перемещение активной рамки происходит нажатием клавиш перемещения курсора, а переход из одного активного окна в другое выполняется нажатием клавиши <Tab>. Отмена действия происходит после нажатия клавиши <Esc>. Распахнуть или уменьшить активное окно можно нажатием <Alt-F5>, закрыть же активное окно, переместиться на более высокий уровень в дереве каталогов - выполняется нажатием <Alt-F4>. И наконец, регенерировать активное окно можно при помощи "горячей" клавиши <F5>. Манипулятор позволяет выполнить все эти действия нажатием левой клавиши "мыши" на соответствующем месте рабочей поверхности и, кроме того, позволяет отмечать группу файлов и каталогов. Для этого надо нажать на левую клавишу, когда стрелка находится на свободном участке активного окна, появится серая рамка, которую можно увеличить в размерах протяжкой "мыши", и после отпускания клавиши все файлы и каталоги, попавшие внутрь этой рамки, будут отмечены. В табл. 1 приведены дейст-

вия, выполняемые оболочкой ViewMAX при нажатии функциональных клавиш.

Оболочка ViewMAX позволяет загружать отмеченные файлы данных нажатием клавиши <Enter> или левой клавиши "мыши" в соответствующую им прикладную программу (по установленной связи прикладная программа - расширение файла данных). Общее число таких связей - не более пятидесяти. В табл. 2 проведено сравнение оболочки ViewMAX с оболочками Norton Commander и Pie Commander.

Оболочка ViewMAX достаточно хорошо защищена от некорректных операций. При выполнении действий, связанных с изменениями файловой структуры (копирование, переименование, стирание), выдается предупреждающее сообщение и пользователю предоставляется альтернатива: продолжить выполнение операции или ее отменить. Кроме того, оболочка всегда контролирует возможность выполнения того или иного действия пункта меню и, если в данный момент этот пункт меню выполнить невозможно, то пункт не подсвечен и доступ к нему запрещен.

Оценивая возможности оболочки ViewMAX, ей можно дать оценку "хорошо". Функции оболочки соответствуют минимуму требований, но в тоже время в нее включены дополнительные средства (см. табл.2). Ограниченное распространение оболочки связано с тем, что установка и настройка оболочки выполняются одновременно с установкой операционной системы DR DOS 5.0. ■

Журнал "Монитор" приглашает к сотрудничеству распространителей печатной продукции

тел. (095) 237-21-36,
(095) 247-36-25

Таблица 1. Действия, выполняемые оболочкой ViewMAX (DR DOS 5.0)

Клавиша или сочетание клавиш	Выполняемое действие
<Enter>	выполнение команды, операции, запуск программы и т.п.
<Space>	установка / снятие отметки
<Tab>	смена активного окна
<ESC>	отмена предыдущего действия
<F4>	"регенерация" активного окна
<Alt>+<F4>	возврат на старший уровень, закрыть текущий каталог
<Alt>+<F5>	распахнуть / уменьшить активное окно
<Up>, <Down>, <Left>, <Right>	перемещение активной рамки
<Alt>+F	переход к меню "File"
<Alt>+O	переход к меню "Options"
<Alt>+V	переход к меню "View"
<Alt>+M	переход к меню "ViewMAX"
<Alt>+H	переход к меню "Help"
<Alt>+<отмеченная буква меню>	выполнение функции, команды меню
<Ctrl>+T	В режиме просмотра файла: смена способа представления данных

Таблица 2. Сравнение оболочки ViewMAX (DR DOS 5.0) с оболочками Norton Commander и Pie Commander

Параметр	Norton Commander Version 3.0	Pie Commander Version 1.7	ViewMAX (DR DOS 5.0)
Режим	текстовый	текстовый	графический
Объем резидентного модуля, Кбайт	около 13	около 13.5	около 14.7
Число окон	два	четыре	два
Язык сообщений	английский	английский/русский	английский
Установка паролей	отсутствует	отсутствует	защита каталогов и три уровня защиты файлов
Манипулятор "мышь"	поддерживает	поддерживает	поддерживает

Продолжение табл.2.

Параметр	Norton Commander Version 3.0	Pie Commander Version 1.7	ViewMAX (DR DOS 5.0)
Командная строка	да	да	отсутствует
Поиск файлов	да	да	отсутствует
Установка шаблона	да	да	да
Сортировка файлов	да	да	да
Копирование	только файлы	файлы и каталоги	файлы и каталоги
Удаление	только файлы	файлы и каталоги	файлы и каталоги
Перенос	только файлы	файлы и каталоги	отсутствует
Переименование файла или каталога	отсутствует	да	да
Изменение атрибутов файла	возможно	возможно	возможно
Сравнение каталогов	возможно	возможно	отсутствует
Рекурсивный подсчет объема каталогов	отсутствует	возможен	возможен
Просмотр дерева каталогов	да	да	да
Просмотр файлов	текстовые,бинарные, dbf-файлы, файлы электронных таблиц, архивные, pcx-файлы и т.д		текстовые и шестнадцатиричные в коде ASCII
Встроенный редактор	да	да	нет
Создание меню пользователя	глобальное и локальные	глобальное и локальные	нет
Автоматический запуск прикладных программ по расширению файла	возможен	возможен	возможен
Дополнительные возможности	режим обмена с другими компьютерами, электронная почта		работа в сетях, форматирование дискет, встроенные часы-будильник, калькулятор

Снова о вирусах - неформальная рецензия на брошюру П.Л.Хижняка "Пишем вирус и антивирус"

"Не ходите, дети, в Африку гулять!"
К.И.Чуковский. Доктор Айболит.

В последнее время появляется все больше и больше публикаций на вирусную и антивирусную темы. Постепенно формируются группы авторов (да и рецензентов), которые благосклонно относятся к одним коллегам и напропалую поносят других. Автор честно признается, что недолюбливает Н.Н.Безрукова с его безудержной рекламой своей уже устаревшей "Компьютерной вирусологии" и постоянными заявлениями об умственной неполноценности авторов вирусов. Ну да ладно. Именно поэтому у автора не поднимается рука писать рецензии на "Компьютерную вирусологию" и ее близнецов-братьев, вышедших за последние два года в четырех или пяти издательствах.

- Ага! - напишет таинственный Б.Н.Н. в очередном выпуске Софтпанорамы, - на слона Моська тявкать испугалась! Мелковато, господа!

- Да нет, - лениво, ибо неохота тратить силы на бесплодные перепалки, возражает автор. - Просто нет смысла. Ваша реклама уже сидит в печенках у доброй половины всех программистов и пользователей, а следовательно я ничего нового не добавлю.

Все, хватит! Иначе рецензия превратится в воображаемый диалог с воображаемым оппонентом. Читатель уже начинает недовольно хмурить брови, а если автор и в следующем абзаце не вернется к теме статьи, то ты, читатель, зевнешь и перевернешь страницу. Итак, сегодня разговор пойдет о брошюре Петра Львовича Хижняка "Пишем вирус и антивирус", выпущенной издательством ИНТО в конце 1991 года общим тиражом 100 000 экземпляров.

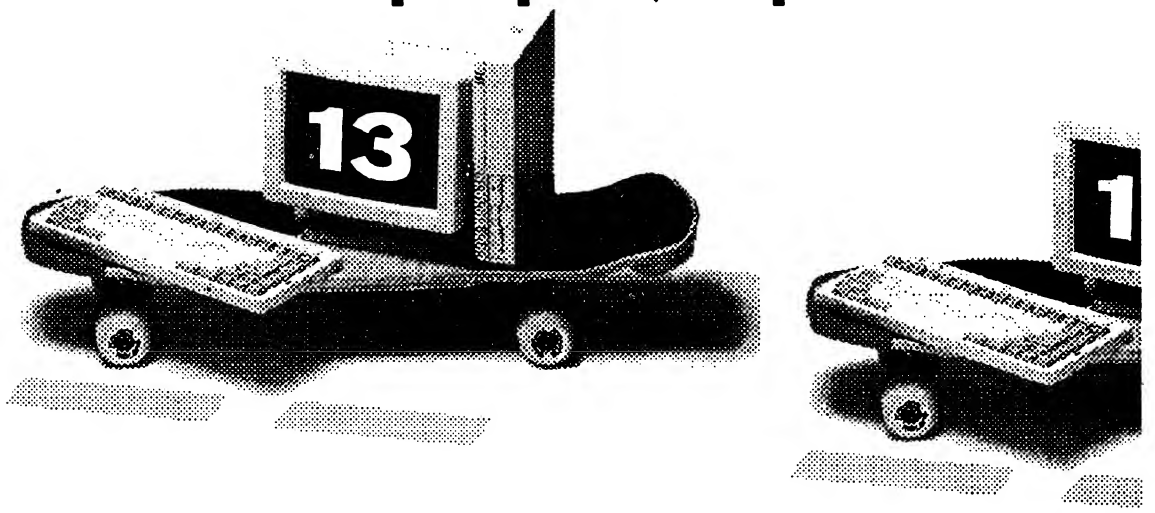
Вначале об авторе. (Далее под автором мы будем понимать уважаемого господина Хижняка П.Л.). К сожалению, не удастся привести его полную биографию, основанную на воспоминаниях его товарищей по детскому саду. Но издатели о нас подумали, и вот чем можно порадовать читателя:

"Петр Львович Хижняк известен советским пользователям IBM-совместимых компьютеров по его книге "Lotus 1-2-3. Справочное руководство" (М., изд. АПН, 1990, 260 с.).

В 1988 г. он читал курс лекций по применению персональных компьютеров во Флоридском государственном университете (США).

стр. 23 ▶

Чемпионат по Windows–программированию



Победил Digitalk – выиграли все

А.Брюзгин

Пользователи больше не хотят тратить недели и месяцы на создание Windows-программ

Специализированные пакеты для создания Windows-программ завоевывают признание у пользователей и специалистов. Их отличают объектно-ориентированный подход к программированию, широкие графические возможности, удобные высокоуровневые интерфейсы баз данных и протокола DDE. На конференции по Windows и OS/2, проведенной в апреле 1991г. в Сан-Хосе(Калифорния), прошли первые соревнования по Windows- программированию, в которых приняли участие 6 фирм, производящих высокоуровневые средства создания Windows-программ.

Соревнования проходили в два этапа. Сначала участникам предлагалось за ограниченное время (6 часов) написать программу по заданию организаторов. Затем проводилось представление программы участникам конференции и ее тестирование. Всем соревнующимся предлагалось решить следующую задачу.

На основании имеющейся исходной информации каждая команда-участница должна создать Windows-программу, извещающую пользователя о времени и месте проведения мероприятий конференции. Полученные программы должны выполнять следующие задачи:

а) создание и вывод списка мероприятий конференции, отобранных пользователем по какому-либо критерию: тема(темы), автор (авторы), а также вывод времени и места проведения этих мероприятий на плане конференции;

б) вывод списка мероприятий для помещения, выбранного пользователем на плане конференции с помощью мыши;

в) вывод на принтер списка мероприятий и помещений в соответствии с выбором пользователя (для которого могли быть использованы различные ключевые слова).

После тестирования полученных программ зрительское жюри фирмы расположило их в следующем порядке:

Фирма	Продукт
Digitalk	Smalltalk/V Windows
Matesys	ObjectView
Knowledge Garden	KnowledgePro for Windows
MDBS	Object/1
Spinnaker Software	Spinnaker Plus
AimTech	IconAuthor

Smalltalk/V Windows

Пакет фирмы Digitalk продемонстрировал наибольшую общую производительность и эффективность. Windows-версия объектно-ориентированного языка Smalltalk/V появилась в продаже всего за месяц до конференции и соединила в себе многие преимущества уже имеющихся версий для других программных платформ. Из своей более ранней DOS версии пакет Smalltalk/V Windows позаимствовал рациональное и экономное использование аппаратных ресурсов (что крайне важно для Windows-программы с широким использованием графики), а из OS/2 PM версии почерпнул многие привлекательные особенности пользовательского интерфейса, например "живую" прокрутку (списки, которые прокручиваются на экране одновременно с перемещением ползуна линейки прокрутки вместо резкого изменения содержания экрана после остановки ползуна).

Так же как OS/2 и Macintosh версии, пакет Smalltalk/V Windows в значительной степени интегрирован с основной графической системой (Windows), однако при этом обеспечивает совместимость с версиями для других платформ на уровне исходного кода прикладных программ. Объектно-ориентированный подход явно превалирует в исходных кодах программ, написанных на Smalltalk/V. Это может вызвать затруднения у пользователей, привыкших работать с традиционными системами программирования, основанными скорее на процедурах манипуляции данными, а не объектами как таковыми.

Программа, созданная командой Digitalk, полностью удовлетворяла всем требованиям, выдвинутым организаторами соревнований. Пользователь мог выполнять большинство операций с помощью "мыши", выбирая интересующее его место на плане конференции и получая всю необходимую информацию. Дополнительное окно табличного формата использовалось для вывода различной информации о программных продуктах, связанных с теми или иными участниками конференции.

Программа-победитель оставалась на экране в течение всей конференции. И многие ее участники предпочитали пользоваться ею для получения необходимой информации, а не искать ее в отпечатанном справочнике. Несомненно фирма Digitalk после этой конференции приобрела немало новых поклонников.

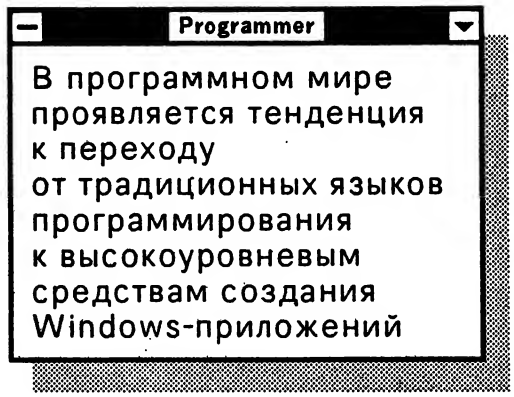
ObjectView

В отличие от всех остальных участников представители фирмы Matesys не воспользовались предоставленным сканированным изображением плана конференции. Они создали подобный план средствами самого пакета ObjectView всего за несколько минут. Кроме того, при создании программы они разделили план на две половины, чтобы укрупнить масштаб и улучшить читаемость картинки. Но именно с точки зрения эффективности использования экрана фирма Matesys существенно уступила фирме Digitalk. Перекрывающиеся и динамически изменяющие размер окна фирмы Matesys производили впечатление большой скорости работы (это, кстати, отразилось и на оценках жюри). Но многочисленные окна в значительной степени перекрывали друг друга. И комбинированные прокручивающиеся окна фирмы Digitalk давали возможность получить больше полезной информации за раз. Программный код, созданный командой Matesys, также был более громоздким, чем код фирмы Digitalk. Интересно, что программы, созданные этими командами, являлись как бы визуальным представлением подходов, реализованных при их разработке. Команда Digitalk использовала интерактивную структуру ООП, а команда Matesys исповедовала более модульный подход. Подход фирмы Digitalk, на мой взгляд, является более предпочтительным при создании специализированных однородных программ. В то же время идеология фирмы Matesys будет иметь преимущества при создании универсальных разнородных приложений.

KnowledgePro for Windows

Занявший третье место пакет KnowledgePro for Windows фирмы Knowledge Garden оказался наиболее чувствительным к проблемам работы с файлами данных.

Эта программа потребовала модификации структуры файла данных путем ввода специального пустого поля в конце каждой записи, для того чтобы удовлетворить требованиям пакета KnowledgePro на специфическое разграничение записей. Необходимость модификации вызвана тем, что описание многострочковых полей в KnowledgePro использует в качестве символов конца строки те, которые обычно разделяют записи в ASCII файлах.



К сожалению, сама эта проблема помешала фирме Knowledge Garden продемонстрировать возможности экспертной системы, заложенные в KnowledgePro. Позже на выставке представитель фирмы Knowledge Garden показал процесс создания системы оперативной подсказки с элементами обучения. Программный код полученного приложения имел несколько необычный вид. Он был разделен на отдельные "темы", связанные друг с другом с помощью матриц связей. Такой подход значительно отличается от общепринятого метода программирования с использованием исполнительных процедур и логических ветвлений. Преимуществами программного подхода фирмы Knowledge Garden являются быстрота создания программ и легкость их отладки. KnowledgePro - прекрасное средство для создания программ, в которых главное - организация сложного многоуровневого взаимодействия программы с пользователем.

Object/1

Четвертый пакет по итогам соревнований, Object/1 фирмы MDBS, оказался самым сложным из представленных на конференции. Object/1 ориентирован скорее на профессионального программиста, а не на опытного любителя. При оценке разработанного командой фирмы MDBS программного кода выяснилось, что разработчики использовали большое количество малопонятных конструкций и даже недокументированных функций.

Так, например, Object/1 выдавал сообщение об ошибке при попытке создать DOS файл с именем, содержащим апостроф. Эта ошибка отражала особенности синтаксиса языка Object/1. При этом программа допускает создание файла с аналогичным именем, использующим символ подчеркивания. Далее Object/1 требует использования в имени файла двойной обратной косой черты, как это принято в языке Си, вместо одной, как делается в DOS. Конечно использование подобных соглашений не имеет особого значения для опытного программиста, но может создать ненужные сложности для обычного пользователя.

Object/1 проявил себя как настоящий пожиратель ресурсов. В результате даже не удалось закрыть программу командой Close, так как не хватило памяти для вывода окна сообщений. Более того, сеанс работы с программой не удалось закончить и с помощью Task Manager. Для этого потребовалось закрыть Windows целиком.

Object/1 предоставляет пользователю мощный интерфейс к различным базам данных и переносимость на OS/2, аналогичную Smalltalk/V. Но это уже задача не для простого пользователя.

Spinnaker Plus

Пакет Spinnaker Plus фирмы Spinnaker Software представляет собой развитие идей, заложенных в программе HyperCard фирмы Apple. Программа, созданная командой фирмы Spinnaker, показала как преимущества, так и недостатки такого подхода к управлению информацией. Индивидуальные экраны программы Spinnaker представляли либо графический план конференции, либо текстовую информацию, но не могли совместить их в одно целое подобно программе фирмы Digitalk. Пакет Spinnaker Plus также не смог обеспечить должный переход от фиксированного формата экрана компьютера Macintosh к более гибкой архитектуре родительских и дочерних окон, которая отличает Windows от более простых графических интерфейсов. Когда основное окно Spinnaker Plus максимизировалось, дочерние окна программы оставались неизменными, а не подстраивались под новый размер родительского окна. Программа содержала нестандартные команды, а также команды, выполнение которых существенно отличалось от принятого в Windows (например, отсутствовала пиктограмма максимизации дочернего окна, а соответствующая команда содержалась в меню). В общем пакет Spinnaker Plus показал себя довольно простым и эффективным средством, слабой стороной которого является совместимость с Windows.

IconAuthor

Пакет IconAuthor фирмы AimTech продемонстрировал на соревнованиях очень резвое начало, и прототип экрана программы появился уже в первый час работы. Однако в дальнейшем команда фирмы AimTech не смогла удовлетворительно справиться с проблемой изменения исходных данных в процессе работы, что и отразилось на окончательном результате. IconAuthor составляет приложения из библиотеки программных единиц, таких как циклы или тесты,

представляя программу на экране в виде диаграммы, индивидуальные блоки которой могут быть открыты и отредактированы с помощью разнообразных средств, выбираемых автоматически IconAuthor в зависимости от типа программной единицы.
Такой подход к созданию приложений делает IconAuthor чрезвычайно удобным для коллективной работы, когда программист занимается организацией последовательности программных единиц и их взаимодействия, а специалист по применению создающейся программы вводит в нее нужную информацию (например, текстовые сообщения или поясняющую графику). IconAuthor предлагает также очень мощную поддержку внешних видеосредств, что делает этот пакет пригодным для создания приложений класса multimedia.

Вместо заключения

Соревнование шести фирм по Windows-программированию, показало существование в программном мире вполне определенной тенденции. По мере того как система Windows становится стандартом пользовательского интерфейса для IBM-совместимых компьютеров, в работу с этой системой включаются все новые и новые пользователи. Причем очень разные пользователи со своими специфическими задачами и проблемами. И многих уже не удовлетворяет состояние дел в области создания прикладных программ для Windows. С одной стороны, пользователи больше не хотят тратить недели и даже месяцы на создание программы и проводить все время за чтением бесконечной документации, скажем, по Windows SDK. С другой стороны, этим пользователям совсем не нужны возможности Windows в полном объеме. Они хотят иметь не Windows-программы вообще, а программы для решения именно их конкретных задач. И вот здесь на программной сцене появились высокоуровневые средства создания Windows-приложений. Почему эти средства называются высокоуровневыми? Дело в том, что они представляют пользователю программный интерфейс Windows не прямо (как это делает SDK или Borland C++), а в виде интерфейса высокого уровня. Например, для того чтобы ввести в программу поддержку протокола DDE, программист, работающий с SDK, должен программировать непосредственный обмен данными с помощью сообщений, поддерживать в программе таблицу атомов для описания данных и т.д. При использовании же программных средств высокого уровня пользователь вводит в свою программу DDE в виде библиотечных функций, не связываясь с самим механизмом реализации протокола. Что конечно существенно упрощает дело.

Характерно, что в соревновании не участвовала ни одна программа традиционного типа (SDK, Borland C++ или Zortech C++). Организаторы поставили перед участниками условия, заведомо обрекающие их на поражение. Действительно попробуйте за шесть часов написать на Си интерфейс к базе данных под Windows, да еще ввести в программу интерактивный графический интерфейс. Да этого не сможет ни один программист в мире!

Правда от участия в соревновании отказались не только производители "классических" пакетов. Не было среди них и таких известных средств разработки Windows-приложений, как Actor фирмы Whitewater и ToolBook фирмы Asymetrix. Как мне кажется, дело здесь в том, что пакет Actor (хотя и является, подобно Smalltalk, самостоятельным объектно-ориентированным языком) по своей организации стоит ближе к SDK, чем к пакетам типа Smalltalk/V Windows. ToolBook же, наоборот, несколько примитивен с программной точки зрения (несмотря на прекрасные графические возможности), и в нем отсутствует поддержка интерфейса баз данных. Следует отметить, что все шесть рассмотренных пакетов не являются универсальными. Они предназначены для решения довольно узкого круга задач. В этом, кстати, и заключается основной недостаток систем высокого уровня. Например, если вам потребуется создать программу, импортирующую графические изображения в каком-нибудь не очень распространенном формате (скажем, в формате пакета CorelDraw), то вы можете это сделать с помощью SDK или другого пакета подобного класса. Программа же высокого уровня позволит вам сделать только то, что в нее заложено. И если уж написано, что KnowledgePro импортирует графику только в формате BMP, то тут уж ничего не поделаешь. Придется ждать появления версии пакета, в которую будет включена нужная вам функция.

То же самое можно сказать и об импорте файлов баз данных. Какое же место занимают рассмотренные пакеты в общей гамме средств создания приложений для системы Windows?

На мой взгляд, общую картину программных средств создания Windows-приложений можно представить в виде пирамиды.

Основание пирамиды занимают средства низкого уровня (язык Ассемблер). Ассемблер (или другой аналогичный машинно-ориентированный язык) является наиболее универсальным программным средством не только для Windows, но и для любых других платформ. При этом он является также и наиболее сложным и трудоемким средством. Ассемблер применяется при создании Windows-программ для программирования наиболее ответственных участков больших универсальных программ (типа текстовых процессоров или пакетов художественной графики). Предназначен он исключительно для программистов-профессионалов.

На следующей ступеньке находятся классические языки программирования Си, Си++, Модула и Паскаль, а также языки типа Actor. Средства, ориентированные на эти языки, являются менее универсальными, чем ассемблер (попробуйте, например, написать библиотеку DLL без использования ассемблерного кода). В то же время они предоставляют программисту большие возможности и гораздо более понятный интерфейс Windows. Средства этого класса используются для создания подавляющего большинства коммерческих Windows-программ, поскольку обеспечивают гораздо большую скорость работы, чем средства высокого уровня. Они предназначены в основном для программистов-профессионалов. Однако некоторые пакеты этого класса (например Turbo Pascal for Windows) ориентированы на более широкую аудиторию. В настоящее время наметилась тенденция введения в системы программирования этого класса элементов систем высокого уровня. (Так, фирма Microsoft плачирует выпустить библиотеку DLL с интерфейсом высокого уровня поддержки протокола DDE). Появление элементов высокоуровневого интерфейса конечно существенно

облегчает работу программиста. Но не следует забывать, что все это делается за счет производительности.

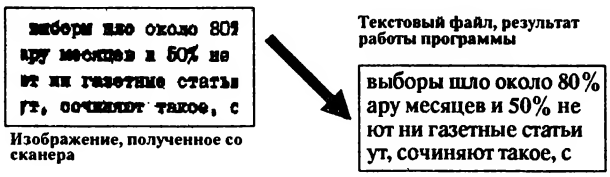
Над традиционными языками программирования находятся средства создания Windows-программ высокого уровня, т.е. как раз те пакеты, о которых мы сегодня говорили. Средства этого класса предназначены для создания программ, работающих в относительно узких областях (например обработки деловой информации). Они включают в себя самые разнообразные компиляторы и интерпретаторы. Этими средствами пользуются программисты, которые создают программы для использования в своей фирме или конторе, чаще всего не предназначенные для коммерческого рынка (такие программы называются "in-house"). Программные средства этого уровня ориентированы в основном на программистов-непрофессионалов, т.е. на тех, для кого программирование не является основной деятельностью. Они имеют, пожалуй, наиболее широкую пользовательскую аудиторию среди всех других, и эта аудитория растет все быстрее. К сожалению, у нас в стране средства этого класса практически отсутствуют. Поэтому программированием для Windows занимаются только профессиональные программисты. А это отсекает от Windows огромную массу достаточно квалифицированных пользователей, которые вынуждены терпеть все неудобства DOS-программирования.

На самом верху пирамиды расположились макроязыки стандартных Windows-программ. К этому классу относятся, например, макроязык пакета Microsoft Excel и язык WordBASIC пакета Microsoft Word for Windows. Средства этого класса обладают еще более узкой областью применения (никакой макроязык не сделает текстовый процессор электронной таблицей и наоборот), но зато предлагают пользователю наиболее удобный и простой программный интерфейс (такие программы называются "off-the-shelf"). Эти макроязыки предназначены для создания на базе основной программы средств, наиболее приспособленных к требованиям конкретного пользователя. Они ориентированы на рядового пользователя, который крайне редко прибегает к какому-либо программированию вообще. И эти средства не находят пока применения у нас, что обусловлено прежде всего преобладанием несанкционированного использования таких пакетов и как следствие отсутствием документации, без которой даже опытному пользователю нелегко работать с незнакомым макроязыком. Но, как мне кажется, преимущества макросредств Windows – программ (в частности, возможность легальной русификации и адаптирования к конкретному пользователю) уже в ближайшем будущем приведут к появлению отечественных разработок именно в этой области. ■

АКЦИОНЕРНОЕ ОБЩЕСТВО
СЕПТИМА SEPTIMA
ПРОГРАММ PROGRAM
СИСТЕМ SYSTEM

AI READER

AI READER избавит Вас от утомительного ручного ввода текстов.
AI READER позволит Вам быстро и надежно ввести в компьютер огромные массивы текста с различных типографских и машинописных источников.



Изображение, полученное со сканера

AI READER, созданная на базе последних достижений в области искусственного интеллекта и распознавания образов, отличается от известных аналогов более высокими показателями надежности и скорости распознавания текста.
AI READER - мобильная и легко адаптируемая система. Вы можете научить
AI READER читать любой новый шрифт с самым сложным начертанием, даже японские иероглифы.

105215, Москва, 11-я парковая, д. 44, корп. 1
Телефоны: 965-3528
965-3529

TVS

Телевизионная охрана объектов внутри и вне помещений

Система TVS - это обнаружение движущихся объектов на контролируемой территории.
Высокая чувствительность, устойчивость к воздействию помех (дождь, снег, качающиеся ветки деревьев и др.) - отличительные черты TVS.
TVS способна обнаружить движущиеся объекты, горизонтальный и вертикальный размер которых составляет менее 1 % от соответствующего размера изображения охраняемого объекта, а контраст не более 75%.
TVS - это комплекс, состоящий из рабочего места оператора: монитора, блока управления системой и набора видеокамер. Все видеоканалы системы работают одновременно и независимо друг от друга. Для каждого канала параметры - зона охраны произвольных размеров, чувствительность, минимальное время обнаружения объекта - настраиваются и фиксируются оператором автономно друг от друга.

UPS

Устройство UPS предназначено для обеспечения бесперебойного питания ПЭВМ типа IBM PC AT/XT.
При возникновении перебоев или при полном отключении питающей сети устройство UPS осуществляет питание нагрузки с помощью автономного источника - аккумуляторной батареи.
Блок UPS обеспечит надежную защиту Вашего компьютера и сохранность программного обеспечения во время сбоя электроэнергии.

Вниманию предприятий, организаций и инофирм!

- **Вы ищете партнера для совместного производства?**
- **Вы заинтересованы в эффективной рекламе Вашей продукции?**
- **Вы хотите получить полную информацию о современном рынке компьютерной техники и программного обеспечения?**

ВЫСТАВКА "ИНФОРМАТИКА-92",

прводимая акционерным обществом "Экспоцентр" при активной поддержке Комитета по информатизации при Министерстве связи России -

- ЭТО ТО, ЧТО ВАМ НУЖНО.

Выставка проводится в Москве на выставочном комплексе на Красной Пресне с 21 по 28 октября 1992 г.

В "Информатике-92" традиционно участвуют крупнейшие фирмы-производители компьютерной техники, программного обеспечения, средств связи, офисного оборудования более чем из 30 стран мира, ведущие отечественные разработчики и производители подобной техники.

Заявки на участие принимаются до 1 августа 1992 года по адресу: 107113, Москва, Сокольнический вал д.1а, телексы 112864, 411185 EXPO SU, факс 288-95-37, телефон 264-90-33

Коммерческий центр Эпсилон

РАБОТАЕМ СТАБИЛЬНО В НЕСТАБИЛЬНОЙ СРЕДЕ

Всегда в наличии:

- *модемы "HAYES"-совместимые, ФАКС-модемные платы;*
- *широкий набор импортных дискет;*
- *чистящие дискеты;*
- *IBM PC AT - 286 и ЗИП к ним;*
- *матричные принтеры;*
- *ленты к принтерам "EPSON", "CITIZEN", "AMSTRAD";*
- *пленка к лазерным принтерам.*

Москва, ул.Пушкинская 7/5, "Дом педагогической книги", 2-й этаж.
Время работы: 10.00-19.00, перерыв 14.00-15.00, кроме воскресенья.
Тел.: в Москве (095) 944-43-65, 943-01-72, 407-69-64, 125-07-67; в Калуге (08422) 4-69-63, 4-54-54.

ГАЗЕТА Человек и компьютер

ПРИГЛАШАЕТ К СОТРУДНИЧЕСТВУ!

Газета "Человек и компьютер" — это самые свежие новости по проблемам информатизации, применения вычислительной техники и программного обеспечения. Газета ставит перед собой задачу стать средством общения программистов и пользователей ЭВМ, быть для них источником интересной и злободневной информации.
Газета "Человек и компьютер" выходит дважды в месяц и распространяется через "Союзпечать" в розницу и по подписке по всем регионам содружества.

Ваш имидж в прессе, Ваша информация, которую необходимо быстро распространить, Ваша реклама – все это успешно может быть решено благодаря постоянному сотрудничеству с газетой "Человек и компьютер".

Если Вы заинтересованы в расширении рынка сбыта компьютерной техники, программных средств и оргтехники, а также многого другого – обращайтесь со своей рекламой в нашу газету.

ТОЛЬКО У НАС: реклама и объявления в короткие сроки и по самым низким в Москве расценкам.

Справки по телефону в Москве: (095) 275-43-36

НАШ ИНДЕКС: 50165

Подписка принимается во всех отделениях связи.
Все ранее вышедшие номера газеты "ЧиК" Вы можете получить в редакции.

Наш адрес: 109280, Москва, Велозаводская, 4
Контактные телефоны: (095) 275-43-36, 275-24-28

Clipper 5.0 – новые возможности

Часть 3. Препроцессор

А. Чернышков

Назначение препроцессоров состоит в предварительной обработке текста программ перед компиляцией. Они могут выполнять следующие функции:
- замену идентификаторов;
- обработку макросов;
- включение в текст других файлов;
- условное включение и/или исключение фрагментов текста;
- расширение языка.

Компилятор языка высокого уровня как правило имеет в своем составе препроцессор (чаще всего недоступный пользователю языка), который осуществляет преобразование синтаксических конструкций языка в синтаксис вызова соответствующих базовых функций, написанных разработчиками языка на языке более низкого уровня. Например, для языка Clipper Summer'87 команда KEYBOARD <символ> преобразовывается препроцессором в функцию__keybord(<символ>), которая содержится в библиотеке CLIPPER.LIB. Препроцессор, встроенный в компилятор, при обработке текста программы создает выходной файл, который затем транслируется в объектный код. Этот файл используется разработчиками языка при его отладке и как правило недоступен пользователю. Тем не менее программист для предварительной обработки текстов программ может пользоваться любым независимым препроцессором, имеющемся на рынке программных продуктов.

Фирма Nantucket Corporation, дав в руки пользователя встроенный препроцессор, сделала еще один шаг к повышению открытости языка. Фактически почти полностью раскрыта реализация языка. Препроцессор Clipper 5.0 позволяет вносить изменения в грамматику и синтаксис языка, удалять и добавлять новые синтаксические конструкции. Большинство директив препроцессора по синтаксису и назначению аналогичны директивам языка С.

Замена идентификаторов

```
#define BUFFER_SIZE 128
```

Заменяет каждое вхождение идентификатора BUFFER_SIZE в тексте программы на 128.

```
#undef BUFFER_SIZE
```

Отменяет предыдущее определение для идентификатора BUFFER_SIZE.

Макросы (псевдофункции)

```
#define C_YEAR(D) str(year(D),4)
```

Заменяет каждое вхождение выражения C_YEAR(<идентификатор>) на выражение str(year(<идентификатор>),4).

Включение файлов

```
#include "inkey.ch"
```

Заменяет эту строку на содержимое файла inkey.ch.

Условная компиляция

```
#ifndef DEBUG
   set printer to
#else
   set printer to DEBUG.PRN
#endif
```

Если идентификатор DEBUG неопределен, то направить вывод на устройство печати, в противном случае - в файл DEBUG.PRN.

```
#ifdef DEBUG
   memoedit(memoread("DEBUG.PRN"))
#endif
```

Если идентификатор DEBUG определен, то просмотреть файл DEBUG.PRN.

Применение этих директив позволяет улучшить читаемость текста программ, повышает производительность программиста и облегчает модификацию программ. Однако самым мощным средством препроцессора Clipper 5.0 являются директивы #command и #translate.

Директивы #command и #translate

Общий синтаксис записи этих директив:

```
#command    <шаблон-соответствие>  =>  <шаблон-результат>

#translate  <шаблон-соответствие>  =>  <шаблон-результат>
```

Такая запись является правилом, по которому препроцессор преобразовывает входной текст при сопоставлении его с шаблоном-соответствием в выходной текст, соответствующий шаблону-результату.

Рассмотрим сначала директиву #translate как более простую в применении. При трансляции программ, написанных ранее в Clipper Summer'87, сразу же выявилась неполная совместимость языка с его предыдущей версией. Так например, функция, тестирующая изменение данных во время выполнения команды READ, в Clipper Summer'87

Programmer

Большинство директив препроцессора Clipper 5.0 по синтаксису и назначению аналогичны директивам препроцессора языка Си

записывалась UPDATE(), а в Clipper 5.0 - UPDATED(). Таким образом, для перевода ранее написанных программ в Clipper 5.0 необходимо во всех программах заменить написание этой функции. Это можно сделать с помощью контекстного поиска и замены в редакторе текстов, но препроцессор предоставляет более эффективный метод. Создадим в каталоге \CLIPPER5\INCLUDE файл с именем cl87.ch и запишем туда строку

```
#translate UPDATE() => UPDATED()
```

В начале каждого файла .PRG, написанного на Clipper Summer'87, запишем строку:

```
#include "cl87.ch"      && включить файл с заменами функций
                        && Clipper'87 на 5.0
```

Препроцессор проведет соответствующую замену везде, где встретит запись UPDATE() независимо от того, в верхнем или нижнем регистре она записана. Независимость от регистра, в котором записан текст, является одним из отличий #translate от #define. Такую же замену можно было провести, записав в файл cl87.ch строку

```
#define UPDATE() UPDATED()
```

Однако если в тексте встретится запись этой функции хотя бы с одной буквой в нижнем регистре (например UPDATe()), необходимой замены не произойдет.

Другой отличительной особенностью директивы #translate является то, что в шаблоне-результате (т.е. справа от знака "=>") можно записать несколько функций (через символ ";"), например:

```
#translate  Say(<row>,<col>,<expr>,[<color>]) => ;
            SetPos(<row>,<col>) ; DevOut(<expr>,<color>)
```

Этим предложением мы как бы ввели новую функцию Say(), которая реализует последовательный вызов двух функций SetPos() и DevOut(). Таким образом, директива #translate чаще всего используется для создания псевдофункций.

В практическом плане для перевода программ, написанных ранее на Clipper Summer'87, удобнее включить строку

```
#include "cl87.ch"
```

в файл STD.CH, находящийся в каталоге \CLIPPER5\INCLUDE, а не в каждый файл .PRG. Туда же можно записать строки

```
#include "locks87.CH"
#include "time87.CH"
```

которые обеспечат преобразование других функций Clipper Summer'87 в соответствующие функции Clipper 5.0. При трансляции Ваших программ файл STD.CH подключается автоматически.

Выполнение препроцессором директивы #command ничуть не отличается от выполнения директивы #translate. Иными словами везде вместо директивы #translate можно использовать #command и наоборот. Однако рекомендуется применять #translate для создания и переопределения функций, а #command - для создания и переопределения команд.

Рассмотрим в качестве примера определение команды APPEND, описанное в файле STD.CH.

```
#command APPEND [FROM <(file)>] [DELIMITED [WITH <*delim*>];
         [FIELDS <fields,...>]                              ;
         [FOR <for>]                                        ;
         [WHILE <while>]                                    ;
         [NEXT <next>]                                      ;
         [RECORD <rec>]                                     ;
         [<rest:REST>]                                      ;
         [ALL]                                              ;
                                                            ;
      => __dbAppDelim(                                      ;
               <(file)>, <(delim)>, { <(fields)> },         ;
               <{for}>, <{while}>, <next>, <rec>, <.rest.>   ;
               )
```

Препроцессор при обработке текста проверяет соответствие команды шаблону-соответствию и в случае удачи замещает исходную строку на текст, соответствующий шаблону-результату. Шаблон-соответствие состоит из ключевых слов (в примере показаны прописными буквами), маркеров-соответствий (заключены в угловые скобки), литералов (переносятся в выходной текст без изменения) и символов обязательности присутствия в исходном тексте (квадратные скобки). Шаблон-результат в общем случае может состоять из этих же компонентов, а также маркеров-результатов, о которых речь пойдет ниже. Шаблон-соответствие должен начинаться с ключевого слова (имени команды).

Маркер-соответствие в общем случае состоит из идентификатора, по которому осуществляется связь маркера-соответствия с маркером-результатом, ключевых слов и специальных символов, определяющих тип маркера. Маркер-соответствие записывается в угловых скобках (<маркер-соответствие>). Рассмотрим существующие типы маркеров-соответствий на простых примерах, в которых будем пока рассматривать только шаблоны-соответствия.

Регулярный маркер-соответствие

```
#command SAVE SCREEN TO <var>
      => <var> := SaveScreen( 0, 0, Maxrow(), Maxcol() )
```

Маркер, в котором кроме идентификатора больше ничего нет, называется РЕГУЛЯРНЫМ маркером. Препроцессор констатирует соответствие только в том случае, если в исходном тексте на месте <var> стоит одиночный идентификатор. Так, если в тексте программы имеется строка

```
SAVE SCREEN TO A1
```

то после обработки препроцессором она преобразуется в строку

```
A1 := SaveScreen( 0, 0, Maxrow(), Maxcol() )
```

Если же по ошибке была записана строка

```
SAVE SCREEN TO A1,A2
```

то препроцессор сочтет ее не соответствующей шаблону-соответствию и оставит без изменения.

Перечневый маркер-соответствие

```
#command RELEASE <vars,...> => __MXRelease( <"vars"> )
```

Маркер, в котором после идентификатора стоят запятая и три точки называется ПЕРЕЧНЕВЫМ маркером. Препроцессор находит соответствие, если в исходной строке на месте маркера стоит список идентификаторов либо одиночный идентификатор.

Ограниченный маркер-соответствие

```
#command SET CURSOR <x:ON,OFF>
      => SetCursor( If(Upper(<(x)>) == "ON", 1, 0) )
```

Маркер, в котором после идентификатора стоят двоеточие и затем список ключевых слов, называется ОГРАНИЧЕННЫМ маркером. Препроцессор находит соответствие, если в исходной строке на месте маркера стоит одно из слов, указанных в маркере.

Произвольный маркер-соответствие

```
#command ENDIF     <*x*>  => endif
```

Звездочки, окружающие идентификатор маркера, позволят препроцессору определить соответствие шаблону, если на месте маркера в исходном тексте стоит любой текст либо пусто.

Теперь перейдем к рассмотрению маркеров-результатов и будем рассматривать как шаблон-соответствие, так и шаблон-результат.

Регулярный маркер-результат

```
#command RESTORE SCREEN FROM <c>
      => RestScreen( 0, 0, Maxrow(), Maxcol(), <c> )
```

Регулярный маркер-результат записывается точно так же, как и регулярный маркер-соответствие, и переносит входной текст в выходной без изменений.

Простой строкообразующий маркер-результат

```
#command SET COLOR TO [<*spec*>]    => SetColor( #<spec> )
```

Наличие символа # сообщает препроцессору о том, что сопоставленный в маркере-соответствии <*spec*> текст необходимо заключить в кавычки, а при отсутствии текста - поставить двойные кавычки (т.е. преобразовать текст в строковую константу, которая может быть и пустой).

Обычный строкообразующий маркер-результат

```
#command RELEASE <vars,...>    => __MXRelease( <"vars"> )
```

При заключении идентификатора маркера-результата в кавычки сопоставленный входной текст преобразуется в строку и записывается в выходной текст. То есть так же, как и в предыдущем примере. Однако есть существенные отличия. Если входной текст не сопоставлен, то в результирующий текст ничего не запишется. Если же сопоставленный текст является списком, как в нашем примере, то каждый элемент перечня преобразуется в строку и записывается в результирующий текст. Таким образом запись

```
RELEASE A1,A2,A3
```

после обработки препроцессором преобразуется в

```
__MXRelease( "A1", "A2", "A3" )
```

Преобразующий строкообразующий маркер-результат

```
#command SET CURSOR <x:ON,OFF>
      => SetCursor( If(Upper(<(x)>) = "ON", 1, 0) )
```

Если идентификатор маркера-результата заключен в скобки, то сопоставленный входной текст преобразуется в строковую константу только в том случае, когда исходный текст не заключен в круглые скобки. Если входной текст не сопоставлен, то в результирующий текст ничего не записывается . Если сопоставленный текст является списком, то преобразование аналогично обычному строкообразующему маркеру. В качестве иллюстрации рассмотрим следующий исходный текст:

```
SET CURSOR On          && курсор включить
On := "OFF"
SET CURSOR (On)        && курсор выключить
```

После обработки препроцессором получим

```
SetCursor( If(Upper("On") = "ON", 1, 0) )    && курсор включить
On := "OFF"
SetCursor( If(Upper(On) = "ON", 1, 0) )      && курсор выключить
```

Блокообразующий маркер-результат

```
#command SET FILTER TO <xpr> => dbSetFilter( <{xpr}>, <"xpr"> )
```

Идентификатор маркера-результата, заключенный в фигурные скобки, дает указание препроцессору записать сопоставленный входной текст в выходной, как блок кодов без аргументов. Написав в тексте программы

```
SET FILTER TO DATE >= CTOD("01.01.92")
```

после работы препроцессора получим

```
dbSetFilter( {|| DATE >= CTOD("01.01.92")}, "DATE>=CTOD("01.01.92")" )
```

Логический маркер-результат

```
#command RESTORE [FROM <(file)>] [<add: ADDITIVE>]
      => __MRestore( <(file)>, <.add.> )
```

При заключении идентификатора маркера-результата в точки, в результирующий текст записывается значение "истина" (.T.), если входной текст сопоставляется. В противном случае записывается "ложь" (.F.). Сам по себе входной текст в результирующий не записывается. Например, после обработки препроцессором текста

```
RESTORE FROM SCOUP
RESTORE FROM SCREEN ADDITIVE
```

получим

```
__MRestore( SCOUP, .F. )
__MRestore( SCREEN, .T. )
```

После усвоения синтаксиса записи рассмотренных директив все же остаются несколько вопросов.

Вопрос 1. Каким образом вводить в определения команд знаки "меньше", "больше", квадратные и фигурные скобки, если эти символы являются неотъемлемой частью синтаксиса препроцессора ?

Ответ. Перед каждым вышеперечисленным знаком необходимо поставить символ "\".

Пример 1.

Определение команды сортировки числового или символьного двумерного массива по n-му столбцу.

```
#command ASR2 <a> ON <n> => asort(<a>,,,{|x,y| x\[<n>\] < y\[<n>\]})
```

Вопрос 2. Как определить команду, которая реализовала бы возможность с помощью одной этой команды в тексте выполнить список действий?

Ответ. Шаблон-соответствие и шаблон-результат надо повторить дважды, причем повторение заключить в квадратные скобки и изменить идентификаторы маркеров в повторении.

Пример 2.

Определение команды присваивания значений переменным по умолчанию.

```
#command DEFAULT <var1> TO <value1> [, <varN> TO
<valueN> ] ;
        => <var1> := if( <var1> == NIL , <value1> , <var1>) ;
        [; <varN> := if( <varN> == NIL , <valueN> ,<varN>)]
```

Вопрос 3. Как поступить, если в команде с блокообразующим маркером-результатом необходимо передать аргумент в блоке?

Ответ. При написании этой команды в тексте программы описать блок явно.

Пример 3.

Передать объект редактирования (get объект) в функцию контроля вводимых данных (valid в команде @ ... get)

```
nt valid {|get| ValAccount(get)}
```

Сборником примеров по определению команд является файл STD.CH, в котором определены почти все команды Clipper. Пользователь может определить свои команды и поместить в этот же файл. Однако имена новым командам надо придумать такие, чтобы первые 4 буквы не совпадали со стандартными командами Clipper, т.к. препроцессор "распознает" команду, определенную директивами #command и #translate по имени, сокращенному до 4 букв. Однако можно и не обременять себя этим ограничением, если воспользоваться директивами #xcommand и #xtranslate, которые действуют также за исключением того, что не применяется правило четырехбуквенного сокращения.

Если Вам потребуется, чтобы стандартная команда Clipper в Вашей программе выполнялась несколько иначе, то можно ее переопределить, записав определение перед началом текста программы. Препроцессор в этом случае проигнорирует описание этой команды в файле STD.CH и произведет преобразование в соответствии с Вашим новым определением. При этом надо иметь в виду, что действие команды, определенной в тексте программы, распространяется на участок текста от места ее определения до конца файла либо до места ее вторичного переопределения. Так, после текста

```
#command @ <row>, <col> SAY <xpr>
         =>
         DevPos(BASE_ROW + <row>, <col> )
         DevOutPict( <xpr> )
#define BASE_ROW 10
```

вывод на экран по команде @ ... SAY будет осуществляться в строку, номер которой увеличен на 10 относительно номера строки, записанного в этой команде.

Управление препроцессорной обработкой

Директивы препроцессора как правило пишутся непосредственно в тексте программ. Однако имеются возможности для управления через параметры компилятора. Для того чтобы получить текст, являющийся результатом обработки препроцессором Вашего исходного текста, надо выполнить компиляцию, указав параметр /p<имя файла>.

```
clipper <имя файла> /p<имя файла>
```

В этом случае во время компиляции Вашего файла создается файл с расширением .PPO, в котором находится выход препроцессора.

Ввести в текст программы директиву #define можно с помощью параметра /d<идентификатор>[=<текст>]. Этим удобно пользоваться для включения и исключения из объектного модуля участков программы, предназначенных только для отладки. Ограничив эти участки в тексте директивами #ifdef (#ifndef) DEBUG и #endif и компилируя с параметром /dDEBUG, получим программу с отладочным кодом, который исчезает при компиляции в отсутствие этого параметра.

Для включения в список каталогов, используемых при поиске файлов, вводимых в текст по директиве #include, служит параметр /i<путь доступа>.

И наконец, с помощью параметра /u<имя файла> можно заставить препроцессор брать определения команд не из стандартного файла описаний (STD.CH), а из созданного Вами. Таким образом Вы можете сконструировать новый язык для какого-либо конкретного приложения. ■

"ИНФОРМЭЙШН КОМПЬЮТЕР ЭНТЕРПРАЙЗ" (РОССИЯ)
И "ИНТЕРНЕЙШЕНЕЛ ДЕЙТА ГРУП"
(США)
Приглашают Вас на 3-ю Международную выставку

"COMPUTERWORLD FORUM/MOSCOW",

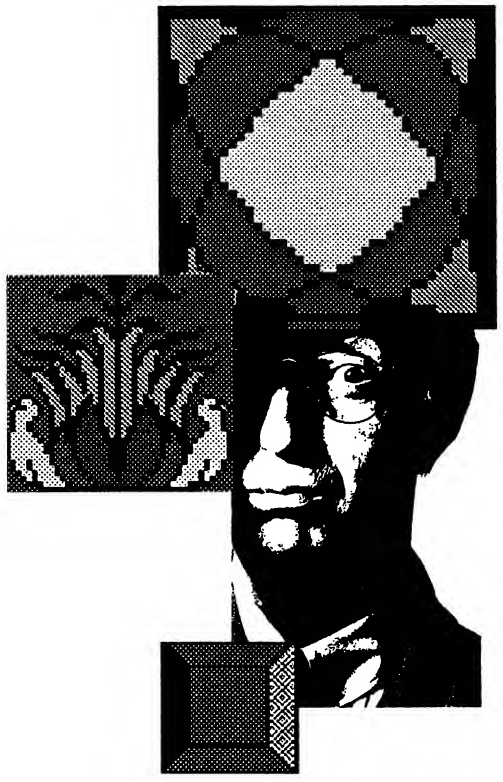
проводимую 17-21 сентября 1992г. в пав. №4 Всероссийского Выставочного Центра

На выставке будут представлены новейшие ПК, периферийные устройства и прикладное программное обеспечение.

Организатор:

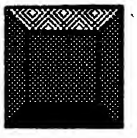
WORLD-EXPO RUSSIA

129223 Москва, проспект Мира, ВВЦ, ПОК, 4 этаж, ком.409.
Телефон: 216-75-90,
Телефакс: 216-83-56.



Показать картинку...

А.Тихонов



Затраты на создание дружественных графических интерфейсов для прикладных программ составляют от 50 до 70 процентов трудоемкости разработки. Если Вы работаете с одним из трансляторов фирмы Borland или Microsoft, то в Вашем распоряжении имеется набор графических примитивов, обеспечивающих установку цвета, отрисовку точек, линий, окружностей, дуг, заливку замкнутых контуров. Однако для отрисовки с их помощью даже таких сравнительно простых интерфейсных объектов, как трехмерных кнопок, движковых регуляторов и циферблатов требуются сотни строк программного текста и десятки часов на отладку.

Выходом является подготовка необходимых фрагментов изображений с помощью графического редактора с тем, чтобы в нужный момент при работе прикладной программы поместить это изображение в определенное место на экране дисплея. Казалось это можно было бы сделать так.

Не мудрствуя лукаво...

В листинге 1. приведена простая процедура на Турбо Паскале (ТП) LoadImage, осуществляющая загрузку из файла и отображение графического образа. LoadImage загружает вначале изображение из файла в буфер, а затем переписывает его в видеопамять. Первое действие осуществляется в три приема: определяется размер буфера с помощью функции ImageSize, затем процедурой GetMet отводится память под буфер, далее загрузка изображения осуществляется с помощью BlockRead. Загруженный образ стандартной процедурой PutImage отображается на дисплее, так что верхний левый угол располагается в точке x1, y1. Параметр PutMode определяет режим отображения: NormalPut - заменяет участок экрана, NotPut делает то же самое, но цвета изображения инвертируются, XorPut, OrPut, AndPut - выполняют с каждым пикселом экрана и загружаемого графического образа сложение по модулю два, логические операции или, и, соответственно. Логические операции выполняются над каждым битом пиксела вне зависимости от того, сколько бит на пиксел использует данный графический режим (1 бит для CGA, 640*200; 2 бита для CGA 320*200, 4 цвета; 4 бита для EGA и VGA, 640*350 и 640*480, 16 цветов, 8 бит, 320*200, 256 цветов).

Однако процедуру LoadImage нельзя непосредственно использовать из-за следующих факторов, основным из которых является отсутствие графического редактора, работающего с графическими образами в формате BGI (Borland Graphic Interface). Другой причиной является то, что LoadImage работает только с изображениями, размер которых не превышает 64 Кбайтов. В то же время для хранения полного экрана CGA необходимы 16 тыс. байтов, EGA - 112 тыс. байтов, VGA - 153,6 тыс.

байтов, Hercules - 31,32 тыс. байтов. Естественно указанное ограничение можно преодолеть, "разрезая" изображение на части и отображая его по частям.

Сколько-нибудь серьезная прикладная программа с графическим интерфейсом использует десятки экранов, так что для их хранения требуются мегабайты дискового пространства (по крайней мере для адаптеров EGA и VGA). Естественным было бы использовать алгоритмы сжатия графической информации. Однако эти алгоритмы должны работать быстро, так что средства, используемые в архиваторах LHARC PKZIP или ARJ, здесь непригодны.

Что же мы хотим

Настало время сформулировать задачу. Необходимо разработать средства загрузки и отображения графических образов, отвечающие перечисленным ниже требованиям:

- подготовка изображений осуществляется с помощью удобного графического редактора или сканера;
- скорость загрузки и вывода изображения должна быть предельно высокой;
- хранение изображения осуществляется в сжатом виде;
- формат хранения изображения должен быть стандартизирован;
- одна и та же программа должна работать с графическими адаптерами CGA, EGA, VGA.

Какой формат выбрать

Графических форматов существует множество: например для ПЭВМ, совместимых с IBM PC, я насчитал их более полутора десятков. Структуры ряда графических форматов приведены в работах [1,2]. Анализ их показал, что наиболее полно поставленной задаче удовлетворяет формат PCX, введенный фирмой ZSoft для своего графического редактора PC PaintBrush и ставший стандартом de facto для графического программного обеспечения, работающего с растровыми изображениями. PCX "понимают" практически все графические редакторы, настольные издательские системы, сканеры.

На базе формата PCX разработан ряд программных инструментов, наиболее известным из которых является PCX Programmer's Toolkit фирмы Genus. К сожалению, версия 3.5 этого пакета не поддерживает Турбо Паскаль версий 5.5 и 6.0, что и явилось одной из побудительных причин настоящей работы.

Что же предлагается

В листинге 2 представлен текст модуля PcxPic, позволяющего загрузить и показать на экране изображение, хранимое в формате PCX в файле (функция ShowPcxFromFile), из библиотеки образов (функция ShowPcxFromLib), имеющей формат библиотеки PCX Programmer's Toolkit. Использование библиотеки целесообразно в том случае, если для работы прикладной программы необходимы более десяти изображений. Сведение их в библиотеку позволяет немного сэкономить дисковое пространство и облегчить сопровождение программы. Затраты на поиск изображения в библиотеке незначительны, но и они могут быть сокращены, если хранить в памяти каталог со смещениями изображений в каталоге, размещенном в оперативной памяти.

Изображения могут быть присоединены к исполняемому модулю редактором связей и находиться в оперативной памяти при выполнении прикладной программы, для чего используется функция ShowPcxFromMemory. Данный способ хранения изображений позволяет увеличить скорость работы программы и рекомендуется, если у Вас медленный винчестер или Вы планируете работать с дискетами.

Перечисленные выше функции имеют единый интерфейс. Левая верхняя граница изображения на экране дисплея определяется координатами x,y. Параметр Gd - код графического драйвера, используется для установки палитры. SetPalette - флажок установки палитры, хранимой вместе с изображением, позволяющий либо установить палитру, либо демонстрировать изображение в текущей палитре. PutMode - режим вывода изображения, может принимать одно из значений NormalPut, NotPut, AndPut, OrPut, XorPut. Строки Name и Lib определяют имена изображения и графической библиотеки. Наконец, P - адрес изображения в оперативной памяти, приредактированного к исполняемому модулю. Все перечисленные функции при успешном выводе изображения на экран дисплея возвращают 0; для их работы необходима установка графического режима процедурой InitGraph.

Контрольный пример вывода изображений из различных источников (TestPcx.PAS) приведен в листинге 3. В примере используются изображения из комплекта поставки PC PaintBrush. Сразу предупреждаю, что изображение должно быть подготовлено для той же графической платы, на которой будет демонстрироваться, иначе Вы рискуете не увидеть ничего кроме пустого экрана. Конверсия изображения может быть выполнена самим графическим редактором PC PaintBrush.

Перейдем к рассмотрению программы TestPcx. В ней вначале устанавливается графический режим и осуществляется сохранение текущей палитры с тем, чтобы восстановить ее в конце работы программы. Затем проводится демонстрация бусинки (BEADS) в различных режимах из файла и графической библиотеки PCL.PCL.

Наконец, для сравнения скорости демонстрации изображений, хранимых на диске и в оперативной памяти, несколько раз вызывается функция ShowPcxFromMemory.

Подредактирование изображения к исполняемому модулю осуществляется в два приема. Во-первых, с помощью утилиты BINOBJ, входящей в комплект поставки Турбо Паскаля, изображение преобразуется в объектный файл. Во-вторых, к исполняемой программе объектный модуль подключается опцией $L (в примере это сделано для изображения {$L BEADS.OBJ}). Кроме этого, объявляется функция с именем этого изображения,

функции ShowPcxFromMemory передается адрес этой процедуры.

Пример заканчивается восстановлением исходной палитры и переходом в алфавитно-цифровой режим. Теперь подошло время разобраться.

Programmer

Эффективная технология создания графического интерфейса включает подготовку фрагментов изображений заранее с помощью графического редактора

Как устроены файлы, создаваемые PC PaintBrush

Графический файл, создаваемый PC PaintBrush, состоит из стандартного заголовка и тела изображения. Формат заголовка PCX файла длиной 128 байтов приводится в листинге 4. В нем хранится информация о типе файла (поле Creator, равное 10, если файл графический), версии (поле Version), типе кодирования изображения (поле Encoding, равное 0, если изображение хранится без сжатия, и 1, если файл был сжат), числе битов, необходимых для кодирования одного пиксела (поле Bits), размерах изображения (поля Xmin, Ymin, Xmax, Ymax), палитре (Palette), количестве цветовых плоскостей (Planes) и количестве байтов, используемых для хранения строки изображения (BytesPerLine).

Тело изображения в формате PCX представляет собой содержимое видеопамяти графического адаптера, а следовательно может меняться для одного и того же изображения при разных адаптерах. Также как и в видеопамяти, изображение разбито на строки.

Фирма ZSoft для сжатия видеоинформации использует простой алгоритм RLE (Run Length Encoding - сжатие последовательности одинаковых символов). Изображение рассматривается как последовательность байтов. Одинаковые байты кодируются парой, первый байт (Count) является счетчиком одинаковых байтов, второй— байтом из кодируемой последовательности. Чтобы отличить счетчик, два его старшие бита устанавливаются равными 1. Это позволяет кодировать последовательности длиной не более 63. Байты изображения со значениями больше 191 приходится кодировать двумя байтами, первым из которых является счетчик. Например, одиночный байт с шестнадцатиричным значением FFh, кодируется двумя байтами - C1h, FFh. Последовательность из 32 байтов FFh преобразуется в два байта - E0h, FFh.

RLE дает хорошие результаты для графических адаптеров, использующих при хранении пиксела информации 1 или 2 бита (CGA, Hercules), либо при раздельном хранении цветовых плоскостей (EGA, VGA, 16-цветовые режимы). Степень сжатия образа зависит как от графического адаптера, так и самого изображения ("гладкие" картинки сжимаются сильнее), для CGA и EGA коэффициент сжатия обычно превышает 2. RLE работает неудовлетворительно, когда для представления пиксела необходимо 8 бит (режим 320*200, 256 цветов, VGA), кстати, поэтому он не включен в рассматриваемую программу.

Основным достоинством RLE является простота и скорость декодирования. Действительно для очередного байта B осуществляется проверка, установлены ли старшие биты (B and $C0)<>0, если это так, то данный байт является счетчиком, число повторений равно Count= B and 3F (сбрасываются старшие биты). Следующий за ним байт переписывается в видеопамять в Count экземплярах. Если же старшие биты рассматриваемого байта не установлены, то он переписывается в видеопамять как есть.

Чтение и декодирование графического образа (Readbyte Inc.) осуществляются вспомогательными функциями, приведенными в листинге 5. Чтение байта с декодированием с диска осуществляется функцией ReadPcxByte. Буферизация осуществляется процедурой ReadBlock. Для чтения изображений, не использующих RLE, применяется ReadNonCompressedByte. Такие изображения создаются PC PaintBrush версий моложе 4 при сохранении части изображения.

Больше красок...

Одним из побочных результатов использования изображений, подготовленных графическим редактором, в прикладной программе является возможность использования большего числа цветов, чем располагает Ваш графический адаптер. Это осуществляется, во-первых, за счет использования смешанных цветов, а во-вторых, установки палитры. Последнее касается графических адаптеров EGA и VGA, позволяющих в любой момент демонстрировать 16 из 64 цветов палитры. Работа с палитрой в ТП осуществляется процедурой SetAllPalette.

Настройка цветов в прикладной программе представляет собой достаточно трудную задачу, которую легче всего решить в графическом редакторе, сохранив палитру вместе с изображением с последующей установкой ее в прикладной программе. Для EGA и VGA каждый из 16 входов палитры кодируется 6 битами, по два бита на каждый основной цвет: красный, зеленый, синий. Таким образом, для основного цвета могут быть установлены 4 уровня. Кодировка цветов палитры осуществляется в виде rgbRGB, где r,g,b - младшие биты кодирования красного, зеленого и синего цветов, R, G, B - старшие биты.

Например, вход ноль палитры, соответствующий в умалчиваемой палитре черному цвету, кодируется нулем, серому (красный - 1 (r), зеленый - 1 (g), синий - 1 (b)) соответствует код 56, белому (красный-2 (R), зеленый - 2 (G),синий - 2 (B)) соответствует код 7, ярко белому (красный-3 (rR), зеленый - 3 (gG), синий -3 (bB)), соответствует код 63.

Установка входов палитры мгновенно меняет цветовую гамму и может быть использована для создания цветовых эффектов.

В PCX файле палитра кодируется аналогично, для хранения входа используются все 8 бит байта, так что значение кода цвета лежит в диапазоне от 0 до 255. Перекодировка палитры изображения для установки ее с помощью SetAllPalette осуществляется процедурой SetPcxPalette, представленной в листинге 6.

Для четырехцветного режима CGA информация о палитре хранится в 0-м и 3-м байтах палитры PCX файла, причем в нулевом байте содержится код цвета заднего плана, а в третьем - код палитры. В ТП можно установить одну из четырех палитр с кодами от 0 до 3 (CGAC0, CGAC1, CGAC2, CGAC3). При этом приходится пользоваться стандартной процедурой SetGraphMode, работа которой приводит к очистке экрана.

Как же все это делается?

Текст трех основных функций - ShowPcxFromFile, ShowPcxFromLib и ShowPcxFromMemory, осуществляющих загрузку и демонстрацию изображений в формате PCX из файла, библиотеки и оперативной памяти, приведен в листингах 7,8,9. Эти функции устроены практически одинаково и отличаются только способом чтения изображения: вначале осуществляется инициализация счетчика и ряда других параметров, затем загрузка заголовка PCX и наконец, если это удалось, то построчное чтение изображения и копирование его в видеопамять (процедура ShowPic, приведенная в листинге 10). Вывод изображения осуществляется только в том случае, если оно не выходит за границы экрана. Перечисленные выше функции используют вспомогательные функции чтения заголовка PCX файла (GetPcxHeader), заголовка графической библиотеки (GetLibHeader) и поиска изображения в библиотеке по имени (PcxLibSearch), тексты которых приведены в листинге 11. Форматы заголовков графической библиотеки и ее разделов приводятся в листинге 12. Сопровождение библиотек изображений я осуществляю с помощью диалоговой утилиты, позволяющей создать библиотеку, добавить, заменить, удалить изображения.

Основную работу по выводу изображения выполняет процедура ShowPic (см. листинг 8). Она устанавливает признак типа палитры Screen, настраивается на тип кодирования изображения (параметр Encoding), устанавливает палитру, если это необходимо; в цикле по числу строк в изображении декодирует строки изображения, помещая их в буфер, который динамически размещается в оперативной памяти по указателю PI. Вывод строки на экран осуществляется страндартной процедурой PutImage. ShowPic заканчивается освобождением памяти, занимаемой буфером.

Заключение

Предлагаемая программа естественно может быть усовершенствована прежде всего в направлении ускорения вывода изображений на экран. Это можно сделать, увеличив в ShowPic буфер для изображения. Еще большего ускорения вывода можно достигнуть, переписывая изображение в видеопамять на физическом уровне. Необходимые для этого сведения можно почерпнуть в [3]. Однако в этом случае ShowPic должна быть переписана для каждого используемого графического адаптера. ■

Литература

1. Carlson W.E. A Survey of Computer Graphics Image Encoding and Storage Formats. Computer Graphics, v.25, N'2, 1991.
2. Rimmer S. Bit-Mapped Graphics. Winfcrest Books, Blue Ridge, 1990.
3. Wilton R. Programmer's Guide to PC & PS/2 Video Systems, Microsoft Press, Redmond, 1987.

Начало на стр. 8

Именно там он близко познакомился с компьютерными вирусами и разработал некоторые концепции защиты от вирусов при работе группы пользователей на персональном компьютере."

Красиво звучит! Ну что же, надо полагать, из опуса П.Л.Хижняка можно будет узнать кое-что из того, что он преподавал буржуазным студентам, а также познакомиться с его концепциями защиты от вирусов.

Теперь читаем аннотацию.

"В книге описаны методы создания эффективных антивирусных программ - детекторов и фагов.С этой целью рассмотрены принципы функционирования некоторых типов вирусов и приведен ассемблерный код простейшего COM вируса с подробным рассказом о фазах создания вируса и о работе отдельных его частей.

Приведен также текст антивирусной программы-детектора и фага для данного конкретного вируса. Подробно описан процесс создания программы-антивируса и антивируса в среде MS-DOS для IBM PC-совместимых (орфография сохранена - В.В.В.) компьютеров. В конце книги дана краткая аннотация ряда публикаций по вирусам и антивирусам.

Книга рассчитана на программистов различных уровней подготовки и на пользователей IBM-совместимых компьютеров."

Та-а-ак... О концепциях уже ни слова. И об опыте общения с подрастающими "буржуинами" тоже... Ну да ладно, может что-нибудь есть внутри? (Иначе просто непонятно, зачем эту информацию выносить прямо на обложку!)

Начинаем читать. Из введения мы узнаем, что компьютерами пользоваться опасно, так как уже бывали случаи, что ошибки в программах приводили к аварии ядерного реактора (?), а также предполагается, что компьютер, зараженный вирусом, может запустить ракету с атомной боеголовкой. Далее описаны еще кое-какие страхи, и в конце концов автор ставит перед читателем три вопроса (почти по Чернышевскому, но у того было всего два - кто виноват? и что делать?) :

- откуда взялись компьютерные вирусы?
- что это такое?
- как с ними бороться?

стр. 12

Листинг 1. Загрузка графического образа в формате BGI.

```
Procedure LoadImage(var F:File;x1,y1,PutMode:integer);
    var
        s:word;
        p:pointer;
    begin
        s:=ImageSize(x1,y1,x2,y2);
        GetMem(p,s);
        blockread(F,p^,s);
        PutImage(x1,y1,p^,PutMode);
        FreeMem(p,s);
end;
```

Листинг 2. Модуль PCXPIC.PAS

```
{$R-,S-,I-,V-,D-} {$X+}
{*********************************************************}
{ unit PcxPic -      инструменты для работы с графическими    }
{                    образами в формате PC PaintBrush         }
{ Cohyright (c) 1991, А.Тихонов                               }
{*********************************************************}
unit PcxPic;

Interface
    uses Dos, Graph;
    Function ShowPcxFromFile(x, y, Gd :Integer;
                             SetPalette:Boolean; PutMode:Word;
                             Name:String):Integer;
    Function ShowPcxFromLib(x,y,Gd:Integer;
                             SetPalette:Boolean; PutMode:Word;
                             Lib,Name:String):Integer;
    Function ShowPcxFromMemory(x,y,Gd :Integer;
                             SetPalette:Boolean; PutMode:Word;
                             P: Pointer):Integer;

Implementation

    Const
        PcxBufSize = 4096;
    Type
        FromMode = (FromMemory,FromFile);
        Mem      = array[0..0] of Byte;
        PMem     = ^Mem;
        GetPicByte = Function:Byte;
    var
        F:    File;
        ReadMemoryByte,ReadByte:GetPicByte;

        Count    :word;{ Счетчик }
        W        :Byte;
        Ps       :Word;
        EndFile  :Boolean;{ Флажок конца файла }
        BufSize  :word; { Размер буфера для графического образа }
        PcxBuf   :array[1..PcxBufSize] of Byte; { Буфер }
        Pm       :PMem;

    {$I PCXHDR.INC}
    {$I PCLHDR.INC}
    {$I SETPAL.INC}
    {$I AUXPCX.INC}
    {$I READBYTE.INC}
    {$I SHOWPIC.INC}
    {$I FROMLIB.INC}
    {$I FROMFILE.INC}
    {$I FROMMEM.INC}
end.
```

Листинг 3. Контрольный пример (TESTPCX.PAS)

```
{$R-,$S-,$I-,$D-,$V-,$B-}
Program TestPcx;
    uses Crt,Graph,PcxPic;

    var
        S:String;
        c:char;
        gm,gd,g,i,j:integer;
        Name,Lib,Image:STRING;
        P:PaletteType;
        Ptr:Pointer;

{$L Beads.obj} {графический образ,
                                    приредактируемый к программе}

        Function Beads:Byte;External; {связывание графического
                                    образа с функцией}
begin
    DetectGraph(gd,gm);
    InitGraph(Gd,Gm,'e:\tp6\bgi'); {установка графического режима}
    if GraphResult0 then begin
        writeln('Ошибка при инициализации графики:',GraphResult);
        halt(0);
    end;
    GetPalette(P);{Сохраняю текущую палитру}
    Lib:='PCL.PCL';{библиотека образов}
    Image:='BEADS.PCX';{графический образ в файле}

    { Различные режимы вывода изображения, храннимого в файле}

    g:=ShowPcxFromFile(96,0,Gd,True,NormalPut,Image);{из файла}
    Delay(200);
    g:=ShowPcxFromFile(96,0,Gd,True,NotPut,Image);{из файла в
                                                    инверсном режиме}
    Delay(200);
    g:=ShowPcxFromFile(96,0,Gd,True,OrPut,Image);{из файла в
                                                    в режиме "ИЛИ"}
    Delay(200);
    g:=ShowPcxFromFile(96,0,Gd,True,XorPut,Image);{из файла в
                                        режиме "ИСКЛЮЧАЮЩЕЕ ИЛИ" }

    { Демонстрация изображений из графической библиотеки}

    for i:=0 to 10 do begin
        case i of
            0: Image:='BEADS';
            1: Image:='3DCURV';
            2: Image:='3DCURVWT';
            3: Image:='BUTTON';
            4: Image:='3DDOORS';
            5: Image:='BIRDCLR';
            6: Image:='BIRDWT';
            7: Image:='BLUESQ';
            else Image:='3DCURVWT';
        end;

        g:=ShowPcxFromLib(i*48,40+i*10,Gd,
                    True,NormalPut,Lib,Image);
    end;
    { Демонстрация изображения, приредактируемого к }
                { загрузочному модулю }

    Ptr:=@Beads; {адрес изображения}
    for i:=0 to 20 do begin
        {графический образ из памяти}
        g:=ShowPcxFromMemory(i*5,100+i*5,Gd,True,NormalPut,Ptr);
    end;
```

```
    c:=readkey;
    SetAllPalette(P);{восстановление палитры}
    CloseGraph;
end.
```

Листинг 4. Заголовок PCX-файла (PCXHDR.INC)

```
Type
    PcxPalette = array[0..15,0..2] of Byte; {Формат палитры PC PaintBrush}
    {Заголовок PCX-файла}
    PcxHeader = Record  {Заголовок PCX-файла}
        Creator   :    Byte; {ZSoft = 10}
        Version   :    Byte; {Версия PCX:
                              0 - версия 2.5 без палитры;
                              2 - версия 2.8 с палитрой;
                              3 - версия 2.8/3.0 без палитры;
                              5 - версия 3.0 с палитрой.
                             }
        Encoding :     Byte; {1 - тип сжатия RLE}
        Bits      :    Byte; { Число битов на пиксел }
        Xmin,
        Ymin,
        Xmax,
        Ymax           integer;{ Размеры картинки }
        Hres,
        Vres      :    integer;{ Разрешение }
        Palette   :    PcxPalette; {Палитра}
        Vmode     :    Byte; { Резерв}
        Planes    :    Byte; { Число цветовых плоскостей}
        BytesPerLine:  Word; { Число байтов на строку}
        PaletteInfo:   Word; { Информация о палитре:
                              1 - цветная / черно-белая;
                              2 - градации серого;
                             }
        shres,
        svres     :    Word; { Разрешение сканера}
        xtra      :    array[0..53] of Byte;{ Резервировано }
end;
```

Листинг 5. Воспомогательные функции для чтения и декодирование графического образа (READBYTE.INC).

```
    Function ReadPcxByte:Byte; Far;
        var
            tmp:Byte;

        procedure readBlock;
            var
                Result:word;
            begin
                if eof(F) then EndFile:=True
                else begin
                    BufSize:=SizeOf(PcxBuf);
                    blockread(F,PcxBuf,BufSize,Result);
                    if result<BufSize then BufSize:=Result;
                    Ps:=1;
                end;
            end;

        function GetByte:Byte;
            begin
                if EndFile then GetByte:=0
                else begin
                    if Ps = PcxBufSize then ReadBlock
                    else Inc(Ps);
                    GetByte:=PcxBuf[Ps];
                end;
            end;

    begin
        if count>0 then begin
            Dec(count);
            ReadPcxByte:=w;
            Exit;
        end;
        tmp:=GetByte;
        if ((tmp and $C0) = $C0) then begin {счетчик повторений}
            count:= tmp and $3F-1;
            w:=GetByte;
        end
        else begin
            count:=0;
            w:=tmp;
        end;
        ReadPcxByte:= w;
    end;

    Function ReadNonCompressedByte:Byte; Far;

        procedure readBlock;
            var
                Result:word;
            begin
                if eof(F) then EndFile:=True
                else begin
                    BufSize:=SizeOf(PcxBuf);
                    blockread(F,PcxBuf,BufSize,Result);
                    if result<BufSize then BufSize:=Result;
                    Ps:=1;
                end;
            end;

        function GetByte:Byte;
            begin
                if EndFile then GetByte:=0
                else begin
                    if Ps = PcxBufSize then ReadBlock
                    else Inc(Ps);
                    GetByte:=PcxBuf[Ps];
                end;
            end;

    begin
        ReadNoncompressedByte:=GetByte;
    end;

    Function GetByteFromMemory:Byte; Far;
        var
            tmp:Byte;

            function GetByte:Byte;
                begin
                    GetByte:=Pm^[Ps];
                    Inc(Ps);
                end;
    begin
        if count>0 then begin
            Dec(count);
            GetByteFromMemory:=w;
            Exit;
        end;
        tmp:=GetByte;
        if ((tmp and $C0) = $C0) then begin {счетчик повторений}
            count:= tmp and $3F-1;
            w:=GetByte;
```

```
        end
        else begin
            count:=0;
            w:=tmp;
        end;
        GetByteFromMemory:=w;
    end;

    Function GetNonComPressedByteFromMemory:Byte; Far;
    begin
        GetNonComPressedByteFromMemory:=Pm^[Ps];
        Inc(Ps);
    end;
```

Листинг 6. Установка палитры (SETPAL.INC)

```
Procedure SetPcxPalette(Screen:Integer;Pal:PcxPalette);{Установка
                    палитры из заголовка Pcx - образа }
    var
        P:PaletteType;
        I,J,BkCol,CgaPal:Integer;
        c:real;
        B,BB:shortint;

    begin
        GetDefaultPalette(P);{ текущая палитра }
        case Screen of
        0: begin        { EGA, VGA, 16 цветов}
            for I:=0 to P.Size-1 do begin
                B:=0;
                for J:=0 to 2 do begin
                    c:=Pal[I,J];
                    BB:=trunc(c/255.0*3.0);
                    case J of
                        0: begin  { Перекодировка красного цвета}
                                  {в формат rgbRGB}
                            if (BB and 1)>0 then B:=B or $20;
                            if (BB and 2)>0 then B:=B or $04;
                        end;
                        1: begin  { Перекодировка зеленого цвета в}
                                  { формат rgbRGB }
                            if (BB and 1)>0 then B:=B or $10;
                            if (BB and 2)>0 then B:=B or $02;
                        end;
                        2: begin  { Перекодировка синего цвета в }
                                  {формат rgbRGB }
                            if (BB and 1)>0 then B:=B or $08;
                            if (BB and 2)>0 then B:=B or $01;
                        end;
                    end;{case}
                end; {J}
                P.Colors[I]:=B;
            end;{I}
            SetAllPalette(P);
        end;{0}
        2: begin { CGA, 4 цвета }
            BkCol:=(Pal[0,0] div $10) mod 8;
            CgaPal:=(Pal[1,0] div $20) mod 4;
            SetBkColor(BkCol);
            SetGraphMode(CgaPal);
        end;
        else;
        end;{case Screen}
    end;
```

Листинг 7. Демонстрация графического образа из файла (FROMFILE.INC)

```
    Function ShowPcxFromFile(x,y,Gd:integer;
                             SetPalette:Boolean; PutMode:Word;
                             Name:String):integer;
    var
        Header:PcxHeader;
        I1,I2,I3,I,J,L:integer;
    begin
        ShowPcxFromFile:=0;
        I1:=OpenFileRead(Name);
        Count:=0;
        EndFile:=False;
        Ps:=PcxBufSize;
        if (I1=0) and (Gd>0) then begin
            I2:=GetPcxHeader(Header);
            if (I2=0) then begin
                if  ((x>=0) and
                    (y>=0) and
                    ((x+Header.Xmax-Header.Xmin)<=GetMaxX) and
                    ((y+Header.Ymax-Header.Ymin)<=GetMaxY)) then
                        ShowPic(FromFile,Header.Encoding,
                            x,y,Gd,SetPalette,PutMode,Header)
                else ShowPcxFromFile:=-2;
            end
            else ShowPcxFromFile:=-1;
        end
        else ShowPcxFromFile:=I1;
        {$I-}
        Close(F);
        {$I+}
    end;
```

Листинг 8. Демонстрация графического образа из библиотеки (FROMLIB.INC)

```
    Function ShowPcxFromLib(x,y,Gd:integer;
                             SetPalette:Boolean; PutMode:Word;
            Lib,Name:String):integer;
    var
        Header:PcxHeader;
        I1,I2,I3,I,J,L:integer;
    begin
        ShowPcxFromLib:=0;
        I1:=PcxLibSearch(Lib,Name,Header);
        if I1=0 then begin
            Count:=0;
            Ps:=PcxBufSize;
            if ((Gd>0) and
                (x>=0) and
                (y>=0) and
                ((x+Header.Xmax-Header.Xmin)<=GetMaxx) and
                ((y+Header.Ymax-Header.Ymin)<=GetMaxY)) then
                    ShowPic(FromFile,Header.Encoding,
                            x,y,Gd,SetPalette,PutMode,Header)
            else ShowPcxFromLib:=-2;
        end
        else ShowPcxFromLib:=-1;

        {$I-}
        Close(F);
        {$I+}
    end;
```

Листинг 9. Демонстрация графического образа, загруженного в исполняемый модуль (FROMMEM.INC)

```
Function ShowPcxFromMemory(x,y,Gd:integer;
                           SetPalette:Boolean; PutMode:Word;
                           P:Pointer):Integer;
var
    Header:PcxHeader;
    I1,I2,I3,I,J,L:integer;
begin
    ShowPcxFromMemory:= 0;
    Pm:= P;
    Count:= 0;
    Ps:= 0;
    if Gd > 0 then begin
        move(Pm^,Header,SizeOf(Header));
        Ps := SizeOf(Header);
        if ((x> =0) and
            (y> =0) and
            ((x+Header.Xmax-Header.Xmin) < =GetMaxX) and
            ((y+Header.Ymax-Header.Ymin) < =GetMaxY)) then
                ShowPic(FromMemory,Header.Encoding,
                        x,y,Gd,SetPalette,PutMode,Header)
        else
            ShowPcxFromMemory:= -2;
    end
    else
        ShowPcxFromMemory:= -1;
end;
```

Листинг 10. Процедура ShowPic (SHOWPIC.INC)

```
Procedure ShowPic(Mode:FromMode;EnCoding,x,y,Gd:integer;
                  SetPalette:Boolean; PutMode:Word;
                  var Header:PcxHeader );
    Type
        ImageMem = record { Участок графического экрана }
        x,y:integer; {Размеры по горизонтали и вертикали в пикселах}
        mem: array[0..0] of Byte;
        end;
    PImageMem =^ImageMem;

    var
        Divisor,BytesPerLine,ImgSize,PixelsPerLine,JJ:Word;
        I,J,L:integer;
        P:pointer;
        PI:PImageMem;
        Screen:integer;
        begin
            case Gd of      { Установка признака палитры}
                EGA,VGA : Screen:= 0;
                HercMono: Screen:= 1;
                CGA     : Screen:= 2;
                else      Screen:= -Gd;
            end;
            if (SetPalette and
                (Header.Version in [2,5])) then { Установка палитры}
                    SetPcxPalette(Screen,Header.Palette);
            If Encoding = 1 then begin
                ReadByte:= ReadPcxByte;
                ReadMemoryByte:= GetByteFromMemory;
            end
            else begin
                ReadByte:= ReadNonComPressedByte;
                ReadMemoryByte:= GetNonCompressedByteFromMemory;
            end;
            Divisor := 8 div Header.Bits;
            PixelsPerLine:= Header.Xmax - Header.Xmin +1;
            BytesPerLine:= PixelsPerLine div Divisor;
            if (PixelsPerLine mod Divisor) < > 0 then Inc(BytesPerLine);
            ImgSize:= ImageSize(Header.Xmin,0,Header.Xmax,0);
        GetMem(PI,ImgSize); { выделяю память под строку изображения}
            { Создание строки образа в формате BGI }
            PI^.x:= PixelsPerLine-1;
            PI^.y:= 0;
            For L:= y to y+Header.Ymax - Header.Ymin do begin { Цикл
                                                  по строкам изображения }
            for j:= Header.Planes - 1 downto 0 do begin { Цикл по
                                                  цветовым плоскостям }
                JJ:= j*BytesPerLine;
                if Mode = FromFile then begin { Заполнение строки
                                                  изображения из файла }
                    for I:= 0 to BytesPerLine-1 do
                        PI^.mem[I+JJ]:= ReadByte;
                    if BytesPerLine < > Header.BytesPerLine then ReadByte
                end
                else begin { Пересылка строки изображения, хранящегося
                                  в оперативной памяти}
                    for I:= 0 to BytesPerLine-1 do
                        PI^.mem[I+JJ]:= ReadMemoryByte;
                    if BytesPerLine < > Header.BytesPerLine then ReadMemoryByte
                end;
            end;
            PutImage(x,L,PI^,PutMode); { Вывод строки изображения на экран}
        end;{L}
        FreeMem(PI,ImgSize);
    end;
```

Листинг 11. Вспомогательные функции (AUXPCX.INC)

```
{Вспомогательные функции}

Function OpenFileRead( Name:String):Integer;
    var
        I:word;
    begin
    {$I-}
        Assign(F,Name);
        I:= Ioresult;
        if I = 0 then begin
            Reset(F,1);
            OpenFileRead:= IoResult;
        end
        else OpenFileRead:= I;
    {$I+}
    end;

Function GetPcxHeader(var Header:PcxHeader):Integer;
    {Чтение заголовок PCX файла}
    var
        s,size:word;
        I:integer;
    begin
        Size:= SizeOf(Header);
        blockread(F,Header,Size,s);
        I:= IoResult;
        If I = 0 then
            if (s = Size) then begin
                if ((Header.version > 5)
                    or (Header.version < 1)
                    or (Header.Creator < > 10)) then GetPcxHeader:= -1
                else GetPcxHeader:= 0;
            end
            else GetPcxHeader:= Size
        else GetPcxHeader:= I;
    end;
```

```
Function GetLibHeader(var Header:PcxLibHeader):integer;
                {Чтение заголовка библиотеки}
const
     pcxLib : array[0..5] of char = ('p','c','x','L','l','b');
var
     s,size:word;
     i,j:integer;
     B: Boolean;
begin
     Size:= SizeOf(Header);
     blockread(F,Header,Size,s);
     i:= IoResult;
     if i = 0 then
          if (s = Size) then begin
               B:= True;
               for j:= 0 to 5 do B:= B and (pcxLib[j] = Header.id[j]);
               if B then GetLibHeader:= 0 else GetLibHeader:= -2;
          end
          else GetLibHeader:= Size
     else GetLibHeader:= i;
end;

Function GetLibDir(var Header:PcxLibDir):integer;
     {Чтение заголовка раздела библиотеки}
     const
          pcxLib : array[0..5] of char = ('p','c','x','L','l','b');
     var
          s,size:word;
          i:integer;
          P: longint;
begin
     Size:= SizeOf(Header);
     blockread(F,Header,Size,s);
     i:= IoResult;
     if i = 0 then
          if (s = Size) then GetLibDir:= 0
          else GetLibDir:= Size
     else GetLibDir:= i;
end;

Function PcxLibSearch( Lib,Name:String;
                       var Header:PcxHeader):integer;
          {поиск раздела Name в библиотеке Pcx - образов Lib}
     const
          pcxLib : array[0..5] of char = ('p','c','x','L','l','b');
     var
          i1,i2,i3,i:integer;
          B:Boolean;
          LH:PcxLibHeader;
          LD:PcxLibDir;
          FP: longint;
     begin
          i1:= OpenFileRead(Lib);
          if i1 = 0 then begin
               B:= GetLibHeader(LH) = 0;
               if B then repeat
                    B:= GetLibDir(LD) = 0;
                    B:= (B and (LD.Synch = 1)); { Проверка флажка
                                        корректности раздела библиотеки}
                    if B then begin
                         for i:= 1 to length(Name) do
                              B:= B and (Name[i] = LD.FileName[i-1]);
                         if not B then begin
                         Fp:= FilePos(F);
                         Fp:= Fp + LD.FileSize;
                         Seek(F,Fp);{Переход к следующему разделу}
                              end;
                         end;
               until (B or eof(F));
               if B then begin
                    B:= GetPcxHeader(Header) = 0;
                    if B then PcxLibSearch:= 0 else PcxLibSearch:= -2;
               end else PcxLibSearch:= -1;
          end
          else PcxLibSearch:= i1;
     end;
```

Листинг 12. Форматы заголовков графической библиотеки и ее разделов (PCLHDR.INC).

```
{Заголовок библиотеки образов }
PCXLIBHEADER = record
     id          : array[0.. 9] of char; {Строка Идентификатор}
     copyright   : array[0..49] of BYTE; {Извещение об авторском праве}
     version     : word;          {Версия}
     pcxlabel    : array[0..39] of char; {Метка тома библиотеки}
     xtra        : array[0..19] of Byte; {Дополнительная информация}
end;

{Заголовок раздела библиотеки образов }

PCXLIBDIR = record
     synch : Byte;                {Флажок корректности раздела,
                                   устанавливается в единицу при
                                   занесении раздела в библиотеку
                                   и в ноль - при удалении или
                                   обновлении}
     filename :array[0..12] of char;{Имя файла изображения}
     filesize :longint:           {Размер файла}
     date : word;                 {Дата}
     time : word;                 {Время}
     pack : word;                 {Тип упаковки}
     note : array[0..39]of char;  {Примечание к изображению}
     xtra : array[0..19]of char;  {Дополнительная информация}
 end;
```

Юнитор-банк

СЭЗ "Янтарь", 236000, г.Калининград (обл.), ул.Земельная 8, фирма "Юнитор", тел. 8-(0112)27-57-78

Операционный день банка (рублевые и валютные счета); сбербанковские операции, синтетические, справочные функции; АРМ председателя правления банка, гл. бухгалтера, ответственного исполнителя, начальника операционного зала. Работает в сети в реальном времени с телекоммуникационной связью с внешним миром.

Необходимы: IBM-совместимые ПК, MS DOS 3.30 и выше, OS-Novell 2.15 и выше.

"Юнитор"

СЭЗ "Янтарь", 236000, г.Калининград (обл.), ул.Земельная 8, фирма "Юнитор", тел. 8-(0112)27-57-78.

Комплексные системотехнические исследования, физико-математическое и экономико-математическое моделирование, структурирование баз данных и баз знаний, оптимизация информационных потоков. Поставка компьютерных аппаратных и программных банковских комплексов.

Novintech

119517,Москва, ул.Нежинская,13,КОНЦЕРН "НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ" ("НОВИНТЕХ")
МП "Системы и Комплексы". Телефоны (095) 441 11 00, (095) 442-57-92
Телефакс (095) 442 57 92

КОНЦЕРН "НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ" ("НОВИНТЕХ"), официальный дистрибьютер и представитель в СССР ведущих фирм мира в области ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (Borland, Microsoft, Lotus, Symantec, Marstek, Nantucket и др.) МП "Системы и Комплексы" ПРЕДЛАГАЕТ:

(Цены указаны по состоянию на 5.05.92 г. и могут быть изменены без уведомления)

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Borland C++ (вер.3.0) & Application Frameworks	28.511 руб
Borland C++ (вер.3.0)	18.830 руб
Turbo C++ for Windows	10.599 руб
Borland C++ Runtime (вер.2.0)	4.972 руб
Turbo C++ (вер.1.0 2nd Edition)	9.335 руб
Turbo Pascal (вер.6.0)	9.146 руб
Turbo Pascal Professional (вер.6.0)	13.475 руб
Turbo Pascal for Windows (вер.1.0)	10.151 руб
Turbo Pascal Runtime (вер.6.0)	6.010 руб
Turbo Pascal for Windows Runtime	6.010 руб
Turbo Debugger & Tools (вер.2.0)	9.379 руб
Turbo C++ & TurboVision	6.164 руб

БИБЛИОТЕКИ ФУНКЦИЙ

Object Professional для Turbo Pascal	43.968 руб
Turbo Professional для Turbo Pascal	27.840 руб
CGT, вер. 1.56 for Clipper 5.0	по запросу
Clipper Tools One for Clipper 5.0	11.520 руб

СУБД

Paradox (вер.3.5)	23.452 руб
Paradox Runtime (вер.3.5)	2.869 руб
Paradox LanPack (вер.3.5x)	7.613 руб
Paradox SQL Link (вер.1.1)	8.452 руб
Paradox SQL Link Rbd/VMS (вер.1.0)	9.294 руб
Paradox Engine (вер.2.0)	8.875 руб
Русификатор для Paradox (версии 3.5x),	5.030 руб
Русификатор для Paradox Runtime	1.220 руб
Русификатор для Paradox LanPack	2.000 руб
Русификатор для Paradox Engine	3.000 руб
dBase IV (вер.1.5)	22.889 руб
dBase IV Lan Pack (вер.1.5)	26.541 руб
dBase IV Developer Edition	39.397 руб
dBase IV (для неоф.польз. в течение 2 · м-цев)	10.845 руб
Clipper, вер.5.01 (Nantucket Corp.)	38.400 руб

ЭЛЕКТРОННЫЕ ТАБЛИЦЫ

QuattroPro (вер.4.0)	9.988 руб
QuattroPro Lan Pack (вер.3.0)	8.304 руб
Framework III (русская версия)	16.332 руб

СКАНЕРЫ (с программным обеспечением)

MERS 800 (Marstec Inc)	449 USD
MARS 105 Plus (Marstek Inc.)	54.592 руб

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА, САПР, РЕДАКТОРЫ, УТИЛИТЫ

ObjectVision (вер.2.0)	7.729 руб
ProtoGen (генерация C и C++ кода)	3.500 руб
SideKick (вер.2.0)	3.137 руб
Norton Commander	3.050 руб
Norton Backup	3.050 руб
Norton Utilities	4.500 руб
Norton Antivirus	3.030 руб
AutoCAD R10 (русская версия)	32.000 руб
AutoCAD R11/286 (русская версия)	51.200 руб
AutoCAD R11/386 (русская версия)	57.600 руб
SEMPRO - Семантический процессор	12.000 руб
MS WORD 5.0 + Русское Слово (вер. 2.0)	12.800 руб
Книга **"Готовим тексты в MS WORD"**	57 руб
Книга **"Работа в Turbo C"**	48 руб

Приобретая программные продукты в Концерне, Вы становитесь зарегистрированным пользователем всемирно известных фирм и получаете право на информационную и техническую поддержку, а также право на приобретение последующих версий программных продуктов со значительной скидкой.

Вы можете стать абонентом Системы Электронной Доски Объявлений по продукции фирмы BORLAND на базе АДС "МИР", которая работает круглосуточно.

МП "Системы и комплексы" Концерна "НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ" ("НОВИНТЕХ"), официальный дистрибьютер корпорации Borland International, предлагает:

dBASE IV v.1.5

- усовершенствованную версию языка dBASE, который является международным стандартом систем управления базами данных и создания приложений.

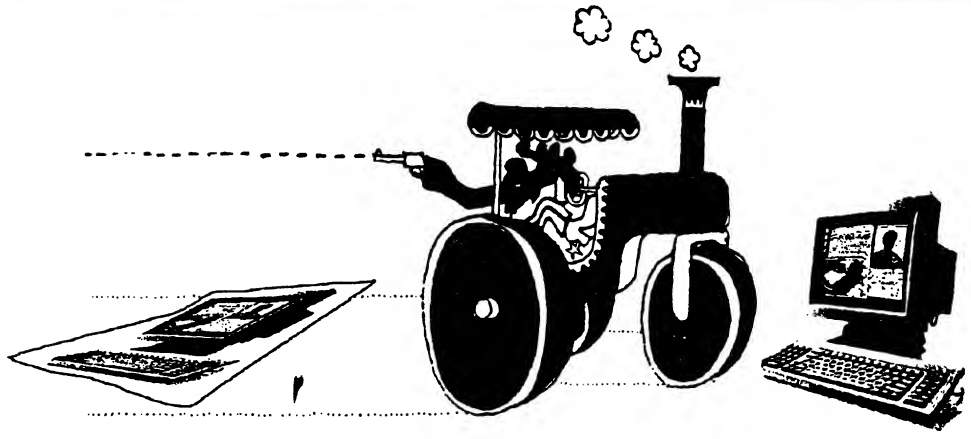
ВНИМАНИЕ! В течение ближайших 2-х месяцев dBASE IV v.1.5 **продается со скидкой 60%** всем **неофициальным** пользователям dBASE, а также официальным пользователям системы Clipper (Nantucket Corp.).

Borland C++ 3.0 и Paradox Engine 2.0

- компилятор C и C++ кода для DOS и Windows вместе с библиотекой процедур и функций C и Pascal для создания баз данных и работы с всемирно известной базой данных Paradox

ВНИМАНИЕ! В течение ближайших двух месяцев при одновременной покупке этих двух продуктов, Вы покупаете их по цене UPGRADE!

12.034 + 4.904 = 16.938 вместо 27.705



Объектно–ориентированный ввод–вывод в Turbo Pascal 6.0. Последовательный доступ

Д. Арапов

Традиционное проектирование структур файлов сводит на нет преимущества, предлагаемые объектно-ориентированным подходом. Однако безвыходных ситуаций не бывает. На помощь приходят потоки ввода-вывода.

Пролог

Почти в каждой большой программе или программной системе есть промежуточные данные, которые необходимо хранить во вторичной памяти чаще всего в виде файла. Разработка системы включает в себя и проектирование формата файла промежуточного представления. В традиционной технологии программирования это один из наиболее ответственных моментов, ибо заложенные в формат файла ограничения могут пережить не одну версию системы (вспомним, скажем, формат .exe файла), порождая у авторов новых версий приступы бессильной ярости: "Ну, где же разместить дескриптор?.." Предусмотрительные разработчики обычно оставляют несколько десятков байтов в заголовке на случай расширения. Такими авторами бывают недовольны дважды. Вначале их критикуют за "рыхлость" и неэффективность формата, а спустя 3-4 версии - за непредусмотрительность и скупость. Изменение формата воспринимается как кощунство, ведь каждый элемент представления "помнит", из какого поля его извлекать, поэтому переделки могут коснуться чуть ли не всей программы.

Цель статьи - показать, что такое объектно-ориентированный ввод-вывод и как он упрощает работу с промежуточными представлениями.

"Не знаю что, не знаю куда...".

В основу объектно-ориентированного подхода легли 3 положения.

1. Пользователь при операции ввода-вывода не обязан знать, с каким именно объектом иерархии он в данный момент имеет дело. Это положение особенно важно при вводе, т.к. иначе потребовалось бы указывать, причем точно, тип еще не прочитанного объекта. Требование "не знаю что" назовем "полиморфностью ввода-вывода по отношению к объекту".

2. Обычно мы имеем дело с файлами DOS, однако иногда промежуточные данные записывают в EMS. В ряде случаев необходимо обеспечить буферизированный ввод-вывод. Удобно, когда в единственной точке программы сказано, с каким носителем идет обмен, а в операциях ввода-вывода пользователь об этом не задумывается. Это требование "не знаю куда" назовем "полиморфностью ввода-вывода по отношению к потоку". А то место, в которое пишут или откуда читают - потоком (Stream).

3. Действия по вводу-выводу объекта определяются методами этого объекта. Централизованное управление (и общая структура файла) отсутствуют. В то же время специфика конкретного типа потока определяется его методами.

TStream и его наследники

В пакете Turbo Vision, входящем в состав системы Turbo Pascal 6.0, реализована схема ввода-

вывода, удовлетворяющая перечисленным требованиям, включая противоречивое требование 3. Рассмотрим этот подход.

Для начала напомним, что все (за несущественными для нас исключениями) объекты в этом пакете - наследники базового объекта TObject. TObject не содержит полей. У него имеется конструктор Init, который обнуляет все поля (в том числе и поля наследников), виртуальный деструктор Done и процедура Free, удаляющая объект.

В число наследников входит и основной герой нашего повествования - объект TStream.

```
type TStream =
    object( TObject )
        Status,Info : Integer;
        procedure Error( Code,Info );virtual;
        procedure Flush; virtual;
        function  Get: PObject;
        function  GetPos: LongInt; virtual;
        function  GetSize: LongInt; virtual;
        procedure Put(P: PObject);
        procedure Read(var Buf; Count: Word); virtual;
        procedure Seek(Pos: LongInt); virtual;
        procedure Truncate; virtual;
        procedure Write(var Buf; Count: Word); virtual;
    end;
```

TStream - абстрактный тип, декларированные в нем виртуальные методы должны быть переопределены в его потомках, реализующих вывод в EMS (расширенную память) или в файл DOS (рис. 1)

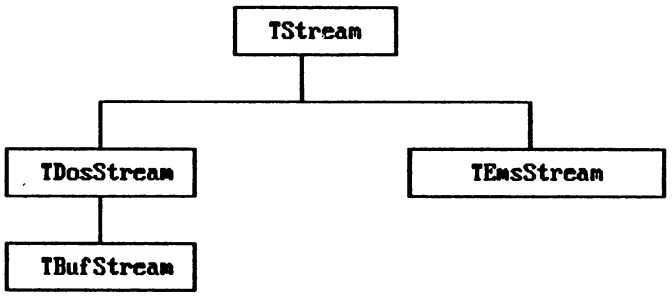


Рис.1 Иерархия потоковых типов в Turbo Vision

Основная роль абстрактных типов состоит в определении интерфейса и тех действий, которые будут выполняться по одному алгоритму для всех потомков (в нашем случае это Put и Get). Создание экземпляров абстрактного типа не предусматривается, поэтому у них обычно отсутствуют конструкторы (по крайней мере собственные).

TEmsStream позволяет хранить промежуточные представления в expanded памяти.

Programmer

Если средний размер объекта невелик, но объектов много, то использование буферизированного потока существенно ускорит операции ввода-вывода

В случае TDosStream поток связан с файлом DOS. В отличие от TStream, который представлял собой абстрактный тип, объекты типа TDosStream создаются и непосредственно применяются в пользовательских программах. Конструктор Init для этого типа имеет 2 параметра: имя файла DOS и режим открытия этого файла (на чтение, запись, создание файла). Процедуры Read и Write читают и пишут в файл, используя средства DOS. Достаточно типичны случаи, когда при чтении или записи необходимо выполнить десятки и сотни обращений к этим процедурам, каждое из которых читает или пишет по 1-2 байта. Чтобы сократить накладные расходы на обращения к операционной системе, удобно использовать тип TBufStream непосредственного потомка TDosStream. DosStream содержит указатель на буфер, а переопределенные в нем виртуальные методы Read, Write и т.д. реализуют буферизированный обмен. Для задания размера буфера служит дополнительный параметр в конструкторе Init.

Покажем, какие необходимы усилия, чтобы вывести в поток, а затем ввести из него некоторый объект. Пусть определены типы

```
PSomeObject = ^TSomeObject;
TSomeObject = object( TObject )
                Field : LongInt;
                ...
                constructor Load ( var S : TStream );
                procedure Store( var S : TStream );
              end;
```

Методы Load и Store определяют, как вводить (выводить) данный тип объектов из (в) произвольный поток, наследующий от TStream.

```
constructor TSomeObject.Load;
    begin       { TSomeObject.Load }
        S.Read( Field,SizeOf(Field) );
    end;        { TSomeObject.Load }
```

Процедура Store пишется аналогично:

```
procedure TSomeObject.Store;
    begin  { TSomeObject.Store }
        S.Write( Field,SizeOf(Field) );
    end;   { TSomeObject.Store }
```

Замечание. Порядок ввода полей из потока в конструкторе Load должен совпадать с порядком их вывода в поток процедурой Store. Поэтому в программах реализацию методов Load и Store очень удобно располагать рядом.

ЗАГС для объектов и потоков

Для того чтобы выводить (вводить) объект в поток, необходимо сообщить системе потокового ввода-вывода некоторую информацию о всех объектных типах, подлежащих обмену. Для каждого такого типа строится константа, имеющая следующий тип записи:

```
const RSomeObject : TStreamRec =
    ( ObjType : word;
    VMTLink : word;
    Load: pointer
    Store: pointer
    );
```

Каково назначение этих полей?

ObjType - уникальный номер типа. Увы, в Turbo Pascal нет механизма, гарантирующего уникальность номера. За нее отвечает пользователь. Некоторые номера (от 0 до 999) зарезервированы фирмой Borland, остальные (от 1 000 до 65 535) можно использовать для своих объектов.

VMTLink - смещение таблицы виртуальных методов для данного объекта. Все таблицы лежат в единственном сегменте данных, поэтому указывать его не требуется.

Load и Store - адреса конструктора Load и процедуры Store (не виртуальной!) регистрируемого типа.

Замечание. Когда тип наследник не определяет новых полей, или по крайней мере таких полей, которые следует сохранять, он может воспользоваться методами Load и Store родительского типа.

При инициализации программы до начала работы с потоками необходимо зарегистрировать вводимый (выводимый) тип(ы) с помощью процедуры RegisterType(R : TStreamRec). Это – обычная процедура (не метод), определенная в модуле Objects.

Вывод динамических структур данных или реальное промежуточное представление

До сих пор не был получен ответ на два вопроса.

1. Кто вызывает методы Load и Store?
2. Как сохранить в потоке динамическую структуру данных?

Рассмотрим систему из двух программ (.exe файлов), которая осуществляет сопоставление текста с шаблоном. Первая программа - это компилятор с языка регулярных выражений в промежуточный код. Вторая программа осуществляет сопоставление текста с заранее скомпилированным шаблоном, который читает из файла промежуточного представления. Структура организации системы приведена на рис. 2.

Разумеется привести текст программ полностью здесь невозможно. В листинге дана общая часть программ, описывающая внутреннее представление шаблонов. Отточиями заменены методы, которые не имеют отношения к описываемой теме.

Компилятор регулярных выражений имеет вид

```
begin
    ...
    Pattern := new( PPattern,Init ));  { Строим дерево разбора }
    RegisterQPattern;                  { Регистрируемся }

    Stream := new( PDosStream,Init( FileName,stOpenWrite ));
    Stream^.Put( Pattern );
    dispose( Stream,Done );
    dispose( Pattern,Done );
end.
```

Программа сопоставления имеет вид

```
begin
    ...
    RegisterQPattern;
    Stream := new( PDosStream,Init( FileName,stOpenRead ));
    Pattern := PPattern(Stream^.Get);
    dispose( Stream,Done );
    ...       { Поиск мы опускаем }
    dispose( Pattern,Done );
end
```

Свое внимание мы сосредоточим на действиях по сохранению в потоке и чтению из него неоднородной структуры данных промежуточного представления. Для начала рассмотрим иерархию типов шаблонов (рис. 3). В корне этой иерархии находится тип TPattern, непосредственный потомок TObject. Это – абстрактный объектный тип. Последнее означает, что в программах не будет создано ни одного объекта этого типа. По этой причине у него нет содержа-

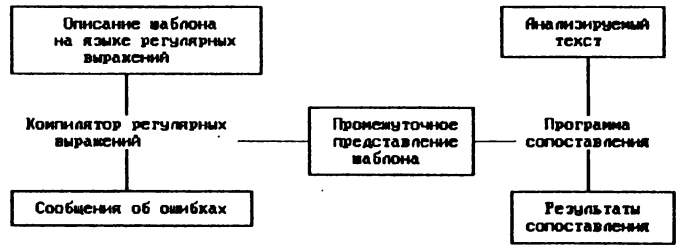


Рис.2. Система программ сопоставления с образцом

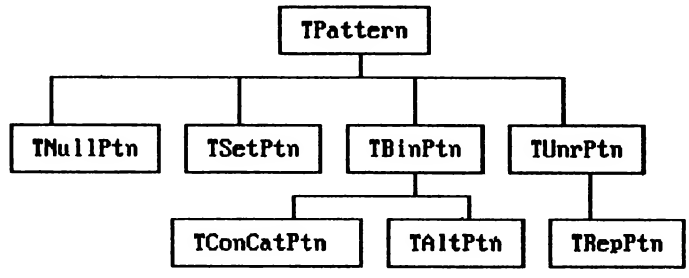


Рис. 3. Иерархия типов шаблонов

тельного конструктора. TPattern служит для выражения абстракции шаблона. Кроме него в иерархию включены еще два абстрактных типа: TUnrPtn и TBinPtn, которые соответственно обобщают понятия унарного и бинарного шаблонов.

В процессе работы порождаются узлы 5 типов:

TNullPtn	- сопоставляется с пустой строкой;
TSetPtn	- сопоставляется с каждым из симоволов множества;
TRepPtn	- повторение подшаблона;
TAltPtn	- один из двух подшаблонов;
TConCatPtn	- последовательность двух подшаблонов.

Шаблон - это дерево. Все нетерминальные узлы этого дерева имеют один из типов: TAltPtn,TConCatPtn или TRepPtn, причем узлы первых двух типов обязаны иметь 2 подшаблона, а третьего - один. Листья дерева принадлежат типам TNullPtn и TSetPtn. Для всех 5 типов написаны регистрационные записи, а процедура RegisterQPattern выполняет регистрацию.

Рассмотрим средства ввода-вывода для каждого из объектов. TPattern не содержит ни собственных полей ни методов. TNullPtn также не имеет полей, но обязан иметь свои методы Load и Store, не выполняющие впрочем никаких действий. TSetPtn содержит методы Load и Store, которые вызывают методы Read и Write потока для чтения и записи единственного поля объекта, задающего множество символов, с которыми отождествляется шаблон. Обратите внимание на использование SizeOf для задания числа считываемых и записываемых байтов. Эта техника упрощает последующее сопровождение программ. Несмотря на то что TUnrPtn и TBinPtn абстрактные типы, они содержат методы Load и Store, которые и упоминаются в регистрационных записях объектов-наследников. У TBinPtn имеются два поля P1 и P2, указывающие на левое и правое поддеревья. Для считывания каждого из них используем метод Get, а для записи соответственно Put. Эти методы выполняют полиморфный ввод-вывод. Одно обращение к ним может повлечь за собой множество обращений к методам ввода-вывода других объектов динамической структуры данных, позволяя ввести или запомнить в потоке произвольную динамическую структуру объектов любых типов. Обратите особое внимание на порядок чтения полей в конструкторе Load. Он должен совпадать с порядком их записи процедурой Store. TUnrPtn реализован аналогичным образом, но содержит только один указатель. У TRepPtn, TAltPtn и TConCatPtn собственных методов Load и Store нет. Они пользуются методами родительских объектов.

Для того чтобы запомнить в потоке целое поддерево шаблонов, содержащее узлы 5 типов, методам достаточно выполнить всего одно обращение Stream^.Put(Pattern). Восстановление дерева из файла также до смешного просто:

```
First := PPattern(Stream^.Get);
```

При этом необходимо только преобразовать указатель от абстрактного типа PObject к нужному нам родовому PPattern. Функция Get и процедура Put служат для полиморфного ввода и вывода объектов. К этим входам обращаются головные программы: компилятор и программа сопоставления.

А все-таки...

А все-таки, спросит дотошный читатель, как представляется объект в потоке? Как TStream узнает тип создаваемого операцией Get объекта?

Для начала рассмотрим логику работы операции Put (рис. 4)

Следует убедиться, что нам не дали NIL. Если передан

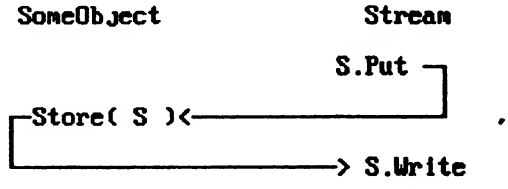


Рис.4. Кто кого вызывает при записи в поток

указатель "в никуда", то с помощью метода Write записываем 0000, т.е. два нулевых байта. На этом считаем свою миссию завершенной.

Если нужно записать настоящий объект, то находим внутри него 2-байтовый указатель на таблицу виртуальных методов. Мы с самого начала предположили, что все рассматриваемые объекты - потомки TObject. У TObject этот указатель всегда находится в невидимом поле со смещением 0.

Заглянем в регистрационные записи и найдем значение ObjType, соответствующее найденному значению в VMTLink. Одна из самых частых ошибок - попытка записи или чтения объекта незарегистрированного типа.

Запишем значение ObjType в поток, используя метод Write.

В регистрационных записях найдем метод Store для данного типа и обратимся к нему. В качестве параметра передаем сам поток.

Функция Get выполняет обратные действия (рис. 5).

Читает 2 байта из текущей позиции потока(при этом используется виртуальный метод Read), если прочитан 0, то возвращаем NIL.

Если ObjType не 0, то ищем в регистрационных записях соответствующий VMTLink.

Ищем в регистрационных записях конструктор Load.

Обращаемся к конструктору Load с указателем Self=NIL

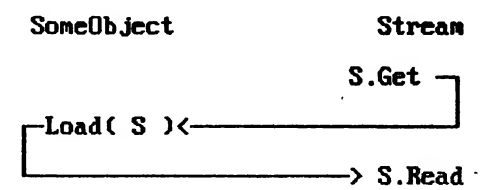


Рис.5. Кто кого вызывает при чтении из потока

SYMANTEC™

SOFTWARE

НОВИНКА
ZORTECH C++ V3.0

оптимизирующий компилятор C++, поддерживающий платформы DOS, OS/2, Windows, 16-bit DOS/16M, 32-bit PharLap DOS Extenders. Extenders включены в комплект поставки. Отладка во всех средах. Стандарт языка AT&T 2.0,2.1. Два варианта поставки: Developers Edition, Science & Engineering Edition. Совместим с Microsoft языками. Библиотеки классов поставляются с исходными текстами. Имеется конвертор программ DOS в Windows. Генератор окон WindowsMAKER. Математические расчеты по стандарту IEEE-754. Версии языка для SCO UNIX V и Macintosh System7.0 поставляются отдельно за рубли.

Time Line 4.0

русифицированная версия одной из самых совершенных систем сетевого планировния и управления проектом. Поддержка локальной сети.

pcANYWHERE IV

популярный коммуникационный пакет, поддерживающий протоколы NetWare/IPX и NetBIOS одобренный.

Peter Norton products

NortonCommander3.0, Norton Editor 2.0, Norton Utilities 6.0, Norton AntiVirus 1.5, Norton BackUp 1.5, Norton Desktop Windows 1.0. Эти и другие продукты для DOS, Windows и Macintosh поставляются за рубли.

ОФИЦИАЛЬНЫЙ ДИЛЕР

121205,Москва,Новый Арбат, д.56,
оффис 1216
Тел.:290-86-30,188-33-10 BBS:290-73-86
Факс:928-07-40
Сетевой адрес
RELCOM:yuri@jet.msk.su

(это знак того, что нужно разместить объект в памяти), с неявным параметром конструктора VMTLink, а в качестве единственного явного параметра передается сам поток.

Однако основная прелесть объектно-ориентированного подхода состоит в том, что мы можем не знать тонкостей работы Put и Get и не задумываться о последовательности байтов в потоке. Если написаны методы Load и Store, построены и зарегистрированы записи типа TStreamRec, то остальные проблемы возьмет на себя система управления потоками.

Мы познакомились с тем, как читать из потока или как записать в поток. При этом рассматривались только последовательная запись или последовательное чтение. При анализе описания TStream можно обнаружить набор операций, не используемых при последовательном доступе (например Seek). Эти процедуры помогают организовать произвольный доступ к потоку. Произвольный доступ организовать значительно сложнее, т.к. объекты (даже одного типа) могут при выводе занимать различное число байтов. О том, как организовать поток "прямого доступа", читайте в следующей статье.

Кулинарные советы

1. Наследуйте от TObject. Потоки могут обрабатывать только наследников этого типа.

2. Пишите и изменяйте методы Load и Store одновременно. Размещайте их в тексте программы друг за другом. Используйте для задания длины поля функцию SizeOf.

3. Если модуль (unit) содержит несколько объектов, которые Вы предполагаете хранить, то создайте процедуру, регистрирующую все объекты этого модуля. Помните, наиболее частая ошибка - попытка прочесть или записать незарегистрированный объект.

4. Если средний размер объекта невелик, но объектов много, то использование буферизированного потока существенно ускорит операции ввода-вывода. ■

Анонс

"Монитор" - версия 4.92

А.Брюзгин. Дух кооперации в Windows: средства межпрограммного обмена данными

Протокол динамического обмена данными (DDE) и средства его поддержки в Windows 3.0+

А.Чернышков. Clipper 5.0 - новые возможности (часть 4)

Подсистема работы с псевдоокнами, иллюстрирующая наиболее интересные программные конструкции языка Clipper 5.0.

С.Крапивенко. Clipper или Paradox: попытка аргументированного выбора (часть 1)

Сопоставляются возможности двух систем, их достоинства и недостатки. Основное внимание уделено эффективности процесса разработки приложений.

А.Штрик. CASE: автоматизированное проектирование программного обеспечения. Концепции и основные положения

Общие принципы построения, классификация и некоторые статистические данные по современным CASE-средствам.

Д.Арапов. Объектно-ориентированный ввод-вывод в Turbo Pascal 6.0. Произвольный доступ

Концепция потоков в Turbo Pascal 6.0. Организация прямого доступа к потокам. Построение баз данных на основе ресурсных файлов.

М.Черкашин. Компилятор пишется так...

На примере простого компилятора, написанного на Паскале, демонстрируются основные приемы, используемые при построении компиляторов. Первая статья в цикле из трех статей.

А.Крапивенко. Программный интерфейс между системой AutoCAD и форматом CLDATA

Описывается формат двоичных файлов обмена чертежами (DXB-файлы), принятый в системе AutoCAD, и международный формат обмена данными CLDATA. Приводится текст программы-перекодировщика чертежей на языке С.

М.Дубицкий. Транспьютеры. Введение в новую концепцию вычислительной техники

Первоначальное знакомство с теоретическими моделями параллельных вычислений и аппаратными средствами, реализующими их в рамках транспьютерной технологии.

И.Василенко, Е.Гулаков. Защищайтесь, сударь! (Часть 1)

Технология создания систем защиты от копирования.

А.Лукашук. Лемминги идут на север

Идея, стратегия, тактика и пароли популярной игрушки "Lemmings Cristmass"

```
unit QPattern;
    interface
        uses Objects;
        procedure RegisterQPattern; { Регистрация всех типов шаблонов }
        type CharSet = Set of Char;
        type PPattern = ^TPattern;   { Абстрактный шаблон }
            TPattern =
                object( TObject )
                ...                  { Методы Load и Store отсутствуют }
                end;
        type PNullPtn = ^TNullPtn;  { Шаблон 'пустая строка'}
            TNullPtn =
                object( TPattern )
                    constructor Init;
                    constructor Load ( var S : TStream );
                    procedure Store( var S : TStream );
                    ...
                end;
        const RNullPtn : TStreamRec =          { Регистрационная запись }
                ( ObjType : 4100;
                VMTLink : Ofs( TypeOf( TNullPtn )^ );
                Load      : @TNullPtn.Load;
                Store     : @TNullPtn.Store );
        type PSetPtn = ^TSetPtn;     { Шаблон сопоставления с множеством
                                       символов }
            TSetPtn =
                object( TPattern )
                    CS : CharSet;         { Множество символов }
                    constructor Load ( var S : TStream );
                    procedure Store( var S : TStream );
                    constructor Init ( C : CharSet );
                    ...
                end;
        const RSetPtn : TStreamRec =
            ( ObjType : 4101;
            VMTLink : Ofs( TypeOf( TSetPtn )^ );
            Load      : @TSetPtn.Load;
            Store     : @TSetPtn.Store );
        type PUnrPtn = ^TUnrPtn; { Псевдо-абстрактный унарный шаблон }
            TUnrPtn =           { Методы Load и Store для всех его }
                object( TPattern )   { наследниках реализованы здесь. }
                    P1 : PPattern;
                    constructor Load ( var S : TStream );
                    procedure Store( var S : TStream );
                    constructor Init( Used : PPattern );
                    destructor Done; virtual;
                    ...
                end;
    { Объектов типа TUnrPtn не создается. Регистрация не нужна.}
        type PRepPtn = ^TRepPtn;    { Шаблон 'повторение P1^' }
            TRepPtn =
                object( TUnrPtn )
                ...
                end;
        const RRepPtn : TStreamRec =
            ( ObjType : 4102;
            VMTLink : Ofs( TypeOf( TRepPtn )^ );
            Load      : @TUnrPtn.Load;
            Store     : @TUnrPtn.Store );
        type PBinPtn = ^TBinPtn; { Псевдо-абстрактный бинарный шаблон,}
            TBinPtn =                { Методы Load и Store для всех его }
                object( TPattern )   {потомков реализованы здесь. }
                    P1,P2 : PPattern;
                    constructor Load ( var S : TStream );
                    procedure Store( var S : TStream );
                    constructor Init( First,Second : PPattern );
                    destructor Done; virtual;
                    ...
                end;
        type PConCatPtn = ^TConCatPtn; { Бинарный шаблон конкатенация.}
            TConCatPtn = object( TBinPtn )
                ...                  { Методы Load и Store унаследованы }
                end;
        const RConCatPtn : TStreamRec =
            ( ObjType : 4103;
            VMTLink : Ofs( TypeOf( TConCatPtn )^ );
            Load      : @TBinPtn.Load;
            Store     : @TBinPtn.Store );
        type PAltPtn = ^TAltPtn;
            TAltPtn =
                object( TBinPtn )
                ...
                end;
        const RAltPtn : TStreamRec =
            ( ObjType : 4104;
            VMTLink : Ofs( TypeOf( TAltPtn )^ );
            Load      : @TBinPtn.Load;
            Store     : @TBinPtn.Store );
    implementation
{************** TNullPtn ************************************}
        constructor TNullPtn.Load;   { У TNullPtn нет полей, которые }
            begin { TNullPtn.Load } { нужно запоминать. }
            end; { TNullPtn.Load }
        procedure TNullPtn.Store;
            begin { TNullPtn.Store }
            end; { TNullPtn.Store }
{************** TSetPtn *************************************}

        constructor TSetPtn.Load;    { В потоке сохраняется множество}
            begin { TSetPtn.Load }   { символов. }
                S.Read( CS,SizeOf(CharSet) );
            end; { TSetPtn.Load }
        procedure TSetPtn.Store;
            begin { TSetPtn.Store }
                S.Write( CS,SizeOf(CharSet) );
            end; { TSetPtn.Store }
        constructor TSetPtn.Init;
            begin { TSetPtn.Init }
                TObject.Init;
                CS := C;
            end; { TSetPtn.Init }
{************** TBinPtn *************************************}
        constructor TBinPtn.Load;    { Чтобы загрузить бинарный }
            begin { TBinPtn.Load }   { шаблон, нужно загрузить }
                P1 := PPattern(S.Get);  { сначала левый, а затем }
                P2 := PPattern(S.Get);  { правый подшаблоны. }
            end; { TBinPtn.Load }
        procedure TBinPtn.Store;     { При работе пользуемся }
            begin { TBinPtn.Store }  { полиморфным вводом-выводом.}
                S.Put( P1 );
                S.Put( P2 );
            end; { TBinPtn.Store }
        constructor TBinPtn.Init;
            begin { TBinPtn.Init }
                TObject.Init;
                P1 := First;
                P2 := Second;
            end; { TBinPtn.Init }
        destructor TBinPtn.Done;
            begin { TBinPtn.Done }
                dispose( P1,Done );
                dispose( P2,Done );
                TObject.Done;
            end; { TBinPtn.Done }
{************** TUnrPtn *************************************}
        constructor TUnrPtn.Load;
            begin { TUnrPtn.Load }
                P1 := PPattern(S.Get);
            end; { TUnrPtn.Load }
        procedure TUnrPtn.Store;
            begin { TUnrPtn.Store }
                S.Put(P1);
            end; { TUnrPtn.Store }
        constructor TUnrPtn.Init;
            begin { TUnrPtn.Init }
                Object.Init;
                P1 := Used;
            end; { TUnrPtn.Init }
        destructor TUnrPtn.Done;
            begin { TUnrPtn.Done }
                dispose( P1,Done );
                TObject.Done;
            end; { TUnrPtn.Done }
        procedure RegisterQPattern;  { Процедура регистрации всех }
            begin { RegisterQPattern }          { типов шаблонов }
                RegisterType( RNullPtn );       { Для каждого типа шаблона, }
                RegisterType( RSetPtn );        { который записывается в }
                RegisterType( RConCatPtn );     { поток, регистрируем запись.}
                RegisterType( RAltPtn );
                RegisterType( RRepPtn );
            end; { RegisterQPattern }
end. { QParttern }
```

Меню как основа технологии взаимодействия с пользователем

С. Курковский

В статье "Технология создания оконного интерфейса" рассказывалось о том, как с помощью языка программирования Турбо Си можно делать дружественные к пользователю программы, как их лучше оформить [1]. Безусловно оконный интерфейс - вещь очень неплохая. Но она станет еще лучше, если в Вашей программе будет какое-нибудь меню. Пользоваться меню куда приятнее, чем мучиться, вспоминая нужные команды для управления программой. Все современные качественные программные продукты в той или иной мере используют системы меню.

Принцип работы с меню заключается в том, что у пользователя есть список возможных вариантов данных для ввода или список возможных действий, из которого нужно выбрать то, что требуется [2].

Мне попадались уже несколько книг по программированию [3,4], в которых очень подробно рассказывалось о том, как создавать системы меню. Но, на мой взгляд, в них есть целый ряд недостатков. Во-первых, в этих книгах авторы стремились как можно меньше использовать библиотечные функции Турбо Си. Во-вторых, описанные системы были достаточно громоздки. В связи с этим родилась идея написать собственную систему меню, которая вобрала бы в себя все лучшее из уже имеющихся, и в то же время была бы более компактной и мобильной.

В описываемой системе происходит построение так называемого POPUP меню. Типичный пример такого меню имеется в популярной оболочке Norton Commander и в любой интегрированной среде программирования фирмы Borland (Integrated Development Environment - языки Турбо Си, Турбо Паскаль и др.). У нас имеется несколько подменю, расположенных по вертикали, которые объединяются в меню более высокого уровня - главное меню, содержащее заголовки этих подменю и располагающееся по горизонтали (в строку). Главное меню будет всегда находиться на экране во время работы основной программы. Вызов главного меню приведет к тому, что работа основной программы будет временно прервана и управление перейдет к системе поддержки и управления меню. Внешне это будет выглядеть так: один из заголовков меню изменит цвет и появится соответствующее ему подменю (какое именно - зависит от параметров, задаваемых главной функции системы). Тогда, передвигаясь по строке главного меню, пользователь может выбрать, какое подменю ему нужно. Выбрав его, пользователь попадает в выбранное подменю и, используя вертикальное перемещение, опять выберет нужный пункт. К этому пункту может быть "привязана" либо какая-нибудь функция, либо новое меню (для этого приведенную систему придется чуть-чуть модифицировать).

Перейдем к более детальному рассмотрению нашей системы. Так как область применения меню очень широка, его конкретная структура зависит от размера и организации меню, от способа выбора его пунктов, возможностей вычислительной системы, реальной потребности пользователя в поддержке со стороны меню и многих других факторов. Наиболее общей и в то же время простой, на мой взгляд, является следующая структура:

```
typedef struct {
        char *name;
        char **item_name;
        char *letters;
        int name_coord;
```

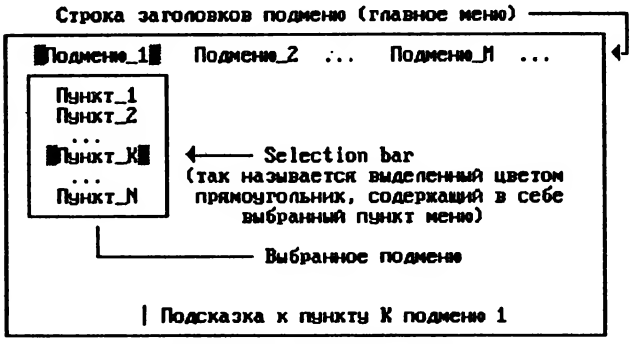


Рис. Вид экрана при работе с POPUP-меню.

```
        int count;
        int width;
        int y;
        int ly;
        int x_coord;
        int y_coord;
        char **perform;
        int **able;
        int (**func)();
} MENU;
```

Эта структура описывает одно подменю. Для описания меню в целом нужно использовать массив таких структур.

Рассмотрим подробно каждую из компонент структуры MENU. Переменная *name - указатель типа char - строка, содержащая заголовок подменю. Массив **item_name - это заголовки подпунктов меню, т.е. строки, которые будут в нем расположены вертикально одна под другой. В каждой строке может появиться символ "~". Это значит, что следующий за ним символ будет выделен другим цветом (сам символ "~" не печатается). Если мы включим все выделенные символы в переменную *letters, то пользователь сможет сразу, не прибегая к передвижению по меню, выбрать соответствующий пункт меню, нажав на клавишу, соответствующую выделенной букве. Такая клавиша обычно называется "hot key". Пусть, например, определена переменная

```
char *item_name[]=
    { " ~Item 1 ",
      " I~tem 2 " };
```

тогда переменная letters должна иметь вид

```
letters="IT";
```

Таким образом, первому пункту меню соответствует hot key "I", а второму - "T".

Переменная name_coord - горизонтальная координата заголовка подменю. Переменная count - количество пунктов данного подменю. В приведенном примере count=2. Переменная width - ширина окна данного подменю.

Переменные y и ly используются для хранения номеров текущего предыдущего пункта меню. Переменные x_coord и y_coord - координаты левого верхнего угла окна с данным подменю.

Массив строк **perform - это подсказки. Они будут выведены на самой нижней строке экрана. Каждому пункту подменю будет соответствовать своя подсказка.

Массив **able - это массив указателей типа int. Если переменная, на которую указывает переменная able соответствующего пункта подменю, равна нулю, значит этот пункт в данный момент недействителен (т.е. не доступен для вызова). Такой пункт пользователь не сможет выбрать - указатель (курсор) не остановится на данном пункте и перейдет на следующий. Также ни к чему не приведет нажатие hot key - оно будет проигнорировано. Существует одно ограничение - в каждом подменю должен быть хотя бы один пункт, для которого соответствующая переменная (на которую указывает able) была бы отлична от нуля. Но это ограничение можно снять, внеся некоторые коррективы.

Последняя оставшаяся переменная - (**func) () - массив указателей на функцию типа int с параметром void (т.е. без параметров). Этой функции будет передано управление, если пользователь выберет соответствующий пункт меню. Если эта функция возвратит ненулевое значение, то управление меню будет прервано и оно исчезнет с экрана, восстановив тем самым первоначальный ход программы (то, что было до вызова меню).

Надо также не забыть глобальную переменную nmenu - в ней записывается общее количество подменю.

Прежде чем перейти к рассмотрению принципа работы функции popup() - главной функции управления меню, приведем краткий обзор вспомогательных функций, вызываемых из popup(), заголовочного файла popup.h и проектного файла popup.prj. Отметим, что для полной программной реализации данной системы понадобятся также функции, описанные в статье [1] (приложение).

Заголовочный файл POPUP.H (листинг 1). В нем содержатся макроопределения всех текстовых атрибутов, использующихся в функциях, описываемых ниже, описание структуры MENU и прототипы всех функций системы.

Функция void status (char *text) (листинг 2). Пишет на нижней строке главную подсказку *text, которая может содержать символы "~". Весь текст, заключенный между двумя такими символами, будет выделен другим цветом.

Функция int snd () (листинг 3). Самая элементарная функция, которая ничего не пишет на экране, но ее работу можно реально проконтролировать.

Функция void perform (char *text) (листинг 4). Похожа на функцию status(), но во-первых, ничего не выделяет цветом, а во-вторых, выводит подсказку *text, немного отступив от левого края строки.

Функция void highlight () (листинг 5). Высвечивает на верхней строке выделенное имя нового подменю.

Функция void fillhead () (листинг 6). Заполняет строку заголовков подменю с помощью функции highlight().

Функция void highput () (листинг 7). Пишет имя пункта подменю в окне. В зависимости от параметра может либо выделить его цветом, либо нет.

Функция void fillmenu () (листинг 8). Используя highput(), заполняет окно подменю.

Функция void move () (листинг 9). Передвигает указатель выделенного пункта подменю.

Теперь подробно опишем работу функции popup() (листинг 10). Эта функция имеет единственный параметр - указатель на массив структур MENU *tmn.

В самом начале функции - вход в цикл while(!done). Глобальные переменные x и lx используются для горизон-

тального перемещения по меню. Если такое перемещение произошло, закрывается окно со списком пунктов подменю, высвечивается новый заголовок подменю. Затем заполняется новое окно с новым подменю.

Следующее условие - если необходимо передвинуться по вертикали (tmn [x].y != tmn[x].ly), то такое передвижение происходит посредством вызова функции move(). Далее происходит обработка нажатых клавиш. Возможные действия (в скобках даны соответствующие определения клавиш из системы TWINDOW [1]):

– передвижение	вверх	(kbup)
	вниз	(kbdown)
	на самый верх	(kbhome)
	на самый низ	(kbend)
	на самое левое подменю	(kbpgup)
	на самое правое подменю	(kbpgdn)
	влево	(kbleft)
	вправо	(kbright)
– вызов соответствующей функции		(kbenter)
– выход		(kbesc)

Кроме того, можно добавить действия, соответствующие непосредственному переходу на необходимое подменю по первой букве его заголовка при нажатой клавише Alt (в приведенных листингах это Alt-A и Alt-B). Если какое-либо действие не попадает в указанный список, проверяется, не совпадает ли значение, возвращенное функцией getevent() из системы TWINDOW, с каким-нибудь из значений переменной letters (напомню, letters - это строка, которая соответствует данному подменю и состоит из символов, которые являются выделенными для каждого из заголовков пунктов подменю, т.е. содержит список всех hot keys для данного подменю). Если да, то будет вызвана соответствующая функция.

Функция main () (листинг 11). В листинге этой функции задаются значения массива структур MENU - заголовки, подсказки, координаты. В самой функции main() подготавливается экран для работы с меню и запускается функция popup().

Проектный файл POPUP.PRJ (листинг 12). поможет Вам скомпилировать всю программу. Имена всех файлов приведены в соответствие с именами функций. Имеется в виду, что, например, функция popup() находится в файле с именем popup.c, а функция main() - в файле main.c.

Теперь несколько слов о том, что нужно сделать, чтобы из подменю можно было вызывать новое меню. Для этого надо воспользоваться рекурсивным вызовом функции popup(), но при этом переменные x, lx и nmenu нужно сделать локальными. Также следует учесть, что подобный вызов "съест" достаточно большое количество памяти.

Возможно, прочитав эту статью и разобравшись в листингах, у Вас появятся идеи по улучшению и оптимизации приведенной системы. Это было бы замечательно - автору будет приятно осознавать, что его труд не пропал даром.

Programmer

Все современные качественные продукты в той или иной степени используют системы меню

Приложение

Система TWINDOW, подробно описанная в статье [1], позволяет быстро и удобно создавать и управлять работой текстовых окон. В TWINDOW входят функции управления курсором, видеопамятью, сохранения и восстановления экрана, динамического управления оконными структурами. Система POPUP в полной мере использует эти возможности. Непосредственно используются четыре функции.

Функция void cursor (void). Используется для изменения формы курсора. В частности для того чтобы сделать его невидимым и для восстановления его предыдущих размеров.

Функция int getevent (void). Является сильно упрощенным фрагментом функции, возвращающей сообщение о последнем событии. В данном случае возвращает скан-код нажатой клавиши.

Функция void closewin (void). Восстанавливает часть экрана, взятую из буфера с помощью библиотечной функции puttext(), уничтожает этот буфер и ненужную теперь структуру хранения оконной информации.

Функция int openwin (). Проверяет, хватит ли свободной памяти для того, чтобы разместить новое окно. Если работа прошла успешно, функция возвращает значение 1, если нет - 0. Приведу подробное описание параметров этой функции.

```
int openwin(x, y, width, height, field, text, frame, title,
            titlestr, framechars, shadow)
    int x, y, width, height;      /* Координаты и размеры окна */
    int field;                    /* Цвета (а не атрибуты) : фона, */
    int text;                     /*                       текста, */
    int frame;                    /*                       рамки, */
    int title;                    /*                    заголовка */
    char *titlestr;       /* Указатель на строку - заголовок */
        /* Если это указатель на пустую строку или NULL, */
        /* Заголовок выведен не будет */
    char *framechars; /* Указатель на строку символов для рамки */
    int shadow;           /* Нужно ли рисовать рамку */
```

■

Литература

1. С. Курковский. Технология создания оконного интерфейса. "Монитор", 1992, №1.
2. Р. Коутс, И. Влеймник. Интерфейс "Человек - Компьютер". М., "Мир", 1990.
3. Herb Schildt. Advanced C. Osborne / McGraw - Hill Book Company. (На русском языке издано "ИВК-Софт" под названием "Язык Си для профессионалов" без указания имени автора).
4. Ал. Стивенс. Современная технология программирования на Турбо Си. (Также издано "ИВК-Софт" без указания имени автора).

Листинг 1. Заголовочный файл POPUP.H

```
#include <conio.h>              /* Используют все функции */

/* Текстовые атрибуты */

/* Обычный пункт меню */
 #define MENUATTR BLACK+16*LIGHTGRAY

/* Выделенная буква обычного пункта меню,
          также - выделенный заголовок подменю */
#define HIGHATTR RED+16*LIGHTGRAY

/* Выделенная буква заголовка подменю */
#define HMATTR   RED+16*CYAN

/* Главная и контекстная подсказка */
#define PERFATTR BLACK+16*CYAN

/* selection bar + обычный пункт меню */
#define NORMATTR YELLOW+16*BLUE

/* Недоступный обычный пункт меню */
#define DISATTR  DARKGRAY+16*LIGHTGRAY

/* selection bar + выделенная буква обычного пункта меню */
#define HINATTR WHITE+16*BLUE

/* Текст главной подсказки */
#define MAINSTATUS \
" ~Д~емонстрация : Нажмите ~ < ~ для тестирования работы системы меню"

typedef struct {                /* Описание структуры MENU */
          char *name;
          char *letters;
          int **able;
          int name_coord;
          int count;
          int width,y,ly;
          int x_coord,y_coord;
     char **item_name;
          int (**func)();
          char **perform;
} MENU;

/*        Прототипы функций для системы POPUP        */
void status(char *s);
int snd();
void perform(char *text);
void highlight(MENU *tmn, int l);
void highput(MENU *tmn, int l);
void fillmenu(MENU *tmn, int x);
void fillhead(MENU *tmn);
void move(MENU *tmn);
void popup(MENU *tmn);
```

Листинг 2. Функция STATUS()

```
/*********************************************/
/*        Функция выводит на нижней строке   */
/*             сообщение главной подсказки   */
/*********************************************/

#include "popup.h"

void status(s) char *s;         /* Строка подсказки */

{
     struct text_info tt;
     int l;

     gettextinfo(&tt);
     window(1, 1, 80, 25);
     gotoxy(1, 25);
     textattr(PERFATTR);
     putch(' ');

     for(l = 0; l < = strlen(s); l++)
     {
          if(s[l] = = '~')          /* Если в строке содержится симол "~" ... */
          {
               l++;
               textattr(HMATTR);
               do
               {
                    putch(s[l]);    /* ... печатать все символы другим цветом ... */
                    l++;
               } while(s[l] ! = '~'); /* ... пока опять не встретится симол "~" ... */
               l++;
               textattr(PERFATTR);
          }
          putch(s[l]);    /* Оставшуюся часть напечатать нормальным цветом */
     }
     clreol();
     window(tt.winleft, tt.wintop, tt.winright, tt.winbottom);
     gotoxy(tt.curx, tt.cury);
}
```

Листинг 3. Функция SND()

```
/*********************************************/
/*        Очень простая функция, которая что-то   */
/*        делает (а именно - очень приятно гудит)  */
/*********************************************/

int snd()

{
     sound(200);
     delay(30);
     nosound();
     return 0;
}
```

Листинг 4. Функция PERFORM()

```
/*********************************************/
/*              Функция выводит подсказку    */
/*                   на нижней строке        */
/*********************************************/

#include "popup.h"

void perform(text) char *text;          /* Строка подсказки */

{
     struct text_info ti;

     gettextinfo(&ti);
     window(1, 1, 80, 25);
     gotoxy(17, 25);
     textattr(PERFATTR);
     cprintf(" %-s", text);
     clreol();
     window(ti.winleft, ti.wintop, ti.winright, ti.winbottom);
     gotoxy(ti.curx, ti.cury);
}
```

Листинг 5. Функция HIGHLIGHT()

```
/*********************************************/
/*          Функция выводит выделенный       */
/*                заголовок меню             */
/*********************************************/

#include "popup.h"

void highlight(tmn, i)
MENU *tmn;          /* Указатель на массив структур MENU */
int i;              /* Номер заголовка */

{
    gotoxy(tmn[i].name_coord, 1);
    textattr(HMATTR);       /* 1-я буква - другим цветом */
    putch((tmn[i].name)[0]);
    textattr(PERFATTR);
    cputs(tmn[i].name + 1);
}
```

Листинг 6. Функция FILLHEAD()

```
/*********************************************/
/*      Функция пишет заголовки всех меню    */
/*********************************************/

#include "popup.h"

extern int nmenu;

void fillhead(tmn) MENU *tmn;   /* Указатель на массив структур MENU */

{
    int i;

    textattr(PERFATTR);     /* Установить нужный текстовый атрибут */
    gotoxy(1, 1);
    clreol();               /* Вывести все заголовки */
    for(i = 0; i < nmenu; i++)
    highlight(tmn, i);
    gotoxy(48, 1);
    cputs(" Это меню написано на TURBO C");
}
```

Листинг 7. Функция HIGHPUT()

```
/*********************************************/
/*      Функция выводит название подпункта   */
/*      меню и контекстную подсказку на      */
/*              нижней строке                */
/*********************************************/

#include "popup.h"

extern int x;

void highput(tmn, i)
MENU *tmn;          /* Указатель на массив структур MENU */
int i;              /* Параметр, отвечающий за выделение цветом */

{
    int pos, i, j, hi = 0;

    switch(i)
    {   case 0 : pos = tmn[x].y;        break;  /* Не выделять */
        case 1 : pos = tmn[x].ly;       break;  /* Не выделять */
        case 2 : pos = tmn[x].y;  hi = 1;  break;  /* Выделить цветом */
    }
    gotoxy(1, pos + 1);
    if(hi) textattr(NORMATTR); else textattr(MENUATTR);
    putch(' ');
    if(*(tmn[x].able[pos])) textattr(MENUATTR);   /* Если меню активно */
        else textattr(DISATTR);
    if(hi) textattr(NORMATTR);          /* Если выделить */

    for(i = 0; i < =strlen(tmn[x].item_name[pos]); i++)
    {
        if((tmn[x].item_name[pos])[i] = ='~')
        {                               /* Этот символ - другим цветом */
            i++;
            if(*( tmn[x].able[pos] )) textattr(HIGHATTR);
            if(hi) textattr(HINATTR);
            break;
        }
        putch((tmn[x].item_name[pos])[i]);
    }

    putch((tmn[x].item_name[pos])[i]);

    if(*(tmn[x].able[pos])) textattr(MENUATTR);
        else textattr(DISATTR);
    if(hi) textattr(NORMATTR);
    cprintf("%-*s", tmn[x].width - i, tmn[x].item_name[pos] + i + 1);
    if(hi) perform( tmn[x].perform[tmn[x].y] ); /* Вывести подсказку */
}
```

Листинг 8. Функция FILLMENU()

```
/*********************************************/
/*      Заполняет окно меню именами подпунктов */
/*********************************************/

#include "popup.h"

void fillmenu(tmn, x)
MENU *tmn;          /* Указатель на массив структур MENU */
int x;              /* Номер текущего подменю */

{
    int i, oldy;

    textattr(MENUATTR);     /* Нужный текстовый атрибут */
    oldy = tmn[x].y;        /* Запомнить старое состояние */
    for(i = 0; i < tmn[x].count; i++)
    {
        gotoxy(1, i + 1);   /* В цикле вывести заголовки */
        tmn[x].y = i;       /* с помощью функции HIGHPUT() */
        highput(tmn, 0);
    }
    tmn[x].y = oldy;        /* Восстановить предыдущий пункт */
}
```

Листинг 9. Функция MOVE()

```
/*********************************************/
/*              Функция передвигает          */
/*              selection bar в окне меню    */
/*********************************************/

#include "popup.h"

extern int x;
```

```
void move(tmn)
MENU *tmn;                    /* Указатель на массив структур MENU */

{
    highput(tmn, 1);    /* Вывести текущий заголовок не выделяя цветом */
    gotoxy(1, tmn[x].y + 1);
    textattr(NORMATTR);
    highput(tmn, 2);          /* Вывести следующий, выделив его цветом */
    tmn[x].ly = tmn[x].y;
}
```

Листинг 10. Функция POPUP()

```
/*****************************************/
/*          Главная функция меню         */
/*****************************************/

#include "popup.h"
#include "twindow.h"

framechars fr = SINGL;        /* Символы для рисования рамок */
            /* (определены в системе TWINDOW - файл TWINDOW.H) */
extern int x, lx, nmenu;      /* см. описание в тексте */

void popup(tmn)
MENU *tmn;                    /* Указатель на массив структур MENU */

{
    int done = 0, begin = 1, n, event;
    struct text_info TI, ti;

    gettextinfo(&TI);
    while(!done)              /* Основной цикл */
    {
        if(x! = lx || begin)  /* Если надо сменить меню */
        {
            window(1, 1, 80, 25);
            if(!begin)
            {
                closewin();
                window(1, 1, 80, 25);
                highlight(tmn, lx);
            }
            else begin = !begin;

        textattr(NORMATTR);
        gotoxy(tmn[x].name_coord, 1);
        cputs(tmn[x].name);
                /* Открыть новое окно */
        openwin(tmn[x].x_coord, tmn[x].y_coord,
                tmn[x].width + 5, tmn[x].count + 2,
                LIGHTGRAY, BLACK, BLACK, BLACK, "", fr, 1);
        gettextinfo(&ti);
        window(ti.winleft, ti.wintop, ti.winright + 1, ti.winbottom);
        /* Перебрать подпункты меню, пока не найдется активное */
        while(!*( tmn[x].able[tmn[x].y] ))
        if(tmn[x].y < tmn[x].count - 1) tmn[x].y + +;
            else tmn[x].y = 0;
        fillmenu(tmn, x);         /* Заполнить новое меню */
        gotoxy(1, tmn[x].y + 1);
        textattr(NORMATTR);
        highput(tmn, 2);
        tmn[x].ly = tmn[x].y;
        lx = x;

        if(tmn[x].y! = tmn[x].ly) move(tmn);
                /* Если надо передвинуть selection bar */
        event = getevent();
        switch(event)
        {
            case kbup     :         /* Двигать вверх */
                do
                {
                    if(tmn[x].y > 0) tmn[x].y--;
                        else tmn[x].y = tmn[x].count-1;
                } while(!*( tmn[x].able[tmn[x].y] ));
                break;

            case kbdown   :         /* Двигать вниз */
                do
                {
                    if(tmn[x].y < tmn[x].count - 1) tmn[x].y + +;
                        else tmn[x].y = 0;
                } while(!*( tmn[x].able[tmn[x].y] ));
                break;

            case kbhome   : tmn[x].y = 0;       /* На верхнюю строку */
                while(!*( tmn[x].able[tmn[x].y] )) tmn[x].y + +;
                break;

            case kbend    : tmn[x].y = tmn[x].count-1; /* На нижнюю строку */
                while(!*( tmn[x].able[tmn[x].y] )) tmn[x].y--;
                break;

            case kbpgup   : x = 0;
                break;          /* На самое последнее меню */

            case kbpgdn   : x = nmenu - 1;
                break;          /* На самое первое меню */

                /* Двигать влево */
            case kbleft   : if(x > 0) x--; else x = nmenu - 1; break;

                /* Двигать вправо */
            case kbright  : if(x < nmenu - 1) x + +; else x = 0; break;

                /* Вызвать соответствующую функцию */
            case kbenter  : if( (*( tmn[x].func[tmn[x].ly] ))() ) return; break;

            case kbesc    : done = 1; break;        /* Выйти */

            case kbalta   : x = 0; break;           /* Меню A */

            case kbaltb   : x = 1; break;           /* Меню B */

            default       : for(n = 0; n < tmn[x].count; n + +)
            if(event = = tmn[x].letters[n] && *(tmn[x].able[n]))
                {               /* Проверить, не совпадает ли нажатая */
                    tmn[x].y = n;   /* клавиша с выделенным символом. */
                    move(tmn); /* Если да, вызвать соответствующую функцию */
                        if( (*( tmn[x].func[tmn[x].ly] ))() ) return;
                    break;
                }
                break;
        }
    }
    closewin();                 /* Убрать меню */
    window(1, 1, 80, 25);
    highlight(tmn, x);
    status(MAINSTATUS);
    window(TI.winleft, TI.wintop, TI.winright, TI.winbottom);
    gotoxy(TI.curx, TI.cury);
}
```

Листинг 11. Функция MAIN()

```
/*************************************************/
/*        Функция демонстрирует возможности      */
/*                системы POPUP                  */
/*************************************************/

#include <alloc.h>
#include "popup.h"
#include "twindow.h"

unsigned long freememory;   /* Количество свободной памяти */
int x, lx,                  /* Номера текущего и предыдущего меню */
    nmenu = 2;              /* Всего 2 меню */
int true = 1, false = 0;

char *a_names[] = {         /* Подпункты меню 1 */
    " ~Item A-A",
    " I~tem A-B",
    " It~em A-C",
    " Ite~m A-D",
    " Item ~A-E",
    " Item A-~F"};

char *a_performs[] = {
    " Это 1-й пункт в меню A",      /* Сообщения - подсказки меню 1 */
    " Это 2-й пункт в меню A",
    " Это 3-й пункт в меню A",
    " Это 4-й пункт в меню A",
    " Это 5-й пункт в меню A",
    " Это 6-й пункт в меню A"};

char a_letters[] = "ITEMAF"; /* Символы, соответствующие пунктам меню 1 */

/* Переменные, отвечающие за доступность соответствующих пунктов меню 1 */
int *a_ables[] = {&true, &true, &true, &true, &true, &true};

/* Указатели на функции, соответствующие пунктам меню 1 */
int (*a_funcs[])() = {snd, snd, snd, snd, snd, snd};

char *b_names[] = {             /* Подпункты меню 2 */
    " ~Item B-A",
    " I~tem B-B",
    " It~em B-C",
    " Ite~m B-D",
    " Item ~B-E",
    " Item B-~F"};

char *b_performs[] = {
    " Это 1-й пункт в меню B",      /* Сообщения - подсказки меню 2 */
    " Это 2-й пункт в меню B",
    " Это 3-й пункт в меню B",
    " Это 4-й пункт в меню B",
    " Это 5-й пункт в меню B",
    " Это 6-й пункт в меню B"};

char b_letters[] = "ITEMBF"; /* Символы, соответствующие пунктам меню 2 */

/* Переменные, отвечающие за доступность соответствующих пунктов меню 2 */
int *b_ables[] = {&true, &true, &true, &true, &true, &true };

/* Указатели на функции, соответствующие пунктам меню 2 */
int (*b_funcs[])() = {snd, snd, snd, snd, snd, snd};

/* Общая структура меню */
MENU tmn [] = {
    {"A - Menu",  a_letters, a_ables, 10, 6, 12, 0, 1, 8, 2,
                  a_names, a_funcs, a_performs},
    {"B - Menu",  b_letters, b_ables, 25, 6, 12, 0, 1, 23, 2,
                  b_names, b_funcs, b_performs}
    };

main()

{
    freememory = MAXMEMSIZE-5000; /* Установим начальный размер */
                                  /* свободной памяти */
    x = 0; lx = 1;              /* Номер текущего меню */
    textattr(NORMATTR);         /* Подготовка рабочего экрана */
    clrscr();
    cursor(false);
    fillhead(tmn);              /* Заголовок меню */
    status(MAINSTATUS);         /* Главная подсказка */
    while(getevent()! = kbenter);       /* Ждать нажатия ENTER */
    popup(tmn);                 /* Вызов главной функции */
    cursor(true);
    textattr(LIGHTGRAY + 16*BLACK);   /* Очистка экрана */
    clrscr();
}
```

Листинг 12. Проектный файл POPUP.PRG

```
main(popup.h, twindow.h)
fillhead(popup.h)
fillmenu(popup.h)
highligh(popup.h)
highput(popup.h)
move(popup.h)
perform(popup.h)
popup(popup.h, twindow.h)
snd
status(popup.h)
getevent(twindow.h)
twindow(twindow.h)
```


INTRALEX Программный комплекс "ЮСИС"

• *Юридическое Информационное Агентство INTRALEX*
107120, г.Москва, пл. Прямикова 5,
Телефон: 272-08-60.
Телекс: 411700 SOCIN 003491
Телефакс: 200-22-17 (200-22-16) SOCIN 003491

• Программный комплекс "ЮСИС" - это юридическая справочно-информационная система; обеспечивает поиск и ввод информации о нормативных актах, научно-практическом комментарии к ним, судебной, арбитражной и нотариальной практике применения, типовых формах договоров, бланков и т.п., включая ведение делопроизводства.

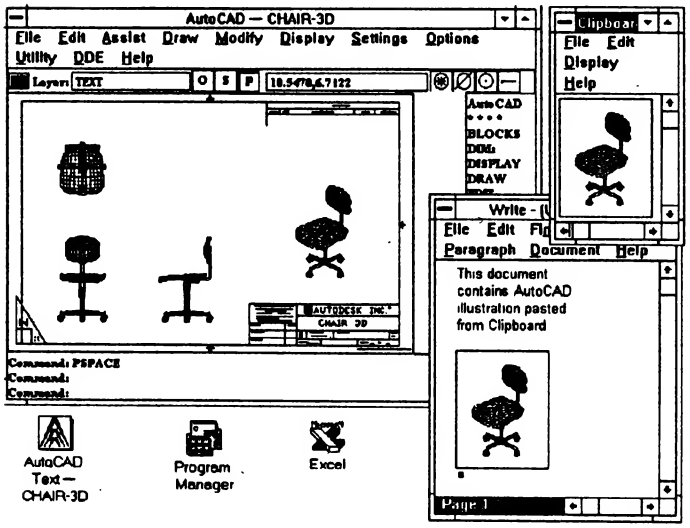
• MS-DOS 3.30 и выше; ПЭВМ совместимые с IBM PC/XT, AT и PS2

• Цена - 14 000 рублей

Журнал "Монитор" приглашает к сотрудничеству распространителей печатной продукции

тел. (095) 237-21-36,
(095) 247-36-25

Система автоматизированного проектирования Автокад версии 11



С. Париевский

Директор по техническому сопровождению Autodesk RF

Данная публикация - введение к серии статей, посвященных вопросам программирования в среде Автокада. Мы планируем опубликовать статьи о написании программ, создающих DXF файлы, об интерфейсе между Автокадом и форматом CLDATA, о программировании на Автолиспе и в Системе разработки приложений для Автокада 386, Автокада для Windows и другие.

На сегодняшний день Автокад версии 11 - самая распространенная в мире система автоматизированного проектирования и черчения. Подобная популярность Автокада на мировом рынке объяснется прежде всего политикой фирмы Autodesk, направленной на непрерывное развитие программных продуктов. Достаточно сказать, что с момента появления в конце 1982 года первой версии Автокада было создано 10 новых версий. Возможность 2- и 3-мерного проектирования и черчения, включая моделирование поверхностей и твердотельных объектов, делает Автокад идеальной системой для широкого ряда приложений. К областям применения Автокада относятся архитектура и строительство, химия, электротехника, электроника и машиностроение; системы автоматизированного управления производством; географические информационные системы; системы планирования, системы настольных типографий и подготовки технических иллюстраций и многие другие.

Благодаря открытой архитектуре системы каждая ее особенность может быть адаптирована для конкретных нужд. Например, можно быстро и легко создать свои собственные меню с наборами применяемых стандартов и часто используемых команд и функций.

Встроенный в Автокад язык программирования Автолисп и новая система разработки приложений (СРП), предусматривающая программирование в среде Автокада на языке Си, дают возможность пользователям и разработчикам прикладных систем создавать как простые, так и очень мощные программы, работающие внутри Автокада для решения практически любых задач проектирования и конструирования.

В настоящее время во всем мире и в нашей стране имеются сотни различных прикладных систем, позволяющие использовать Автокад в разных областях. К ним относятся системы автоматизированного программирования станков с ЧПУ, системы для параметрического проектирования, программы, выполняющие расчеты методом конечных элементов, системы по созданию спецификаций и смет, программы по разводке трубопроводов и кабелей, различные библиотеки стандартных элементов и многие другие.

Автокад поддерживает множество различных устройств ввода-вывода, включая графические мониторы с высоким разрешением, дигитайзеры и "мыши", перьевые и электростатические плоттеры, матричные и лазерные принтеры и даже фотонаборные машины. Уникальный интефейс с периферийными устройствами (Autodesk Device Interface - ADI) позволяет производителям пери-

ферийных устройств легко и просто создавать программы-драйверы для работы своих устройств с Автокадом с учетом их специфических возможностей.

Ниже приводится перечень основных особенностей Автокада версии 11. (Названия всех команд и терминов даются в соответствии с принятыми для Автокада на русском языке.)

Programmer

Автокад версии 11 – самая распространенная в мире система автоматизированного проектирования и черчения

Пространства модели и листа чертежа

Соотношение модели и листа чертежа подчеркнуто разделением среды рисунка на пространство модели, в котором происходит создание объектов, и пространство листа, в котором производится оформление чертежа (рис.1). Механизм пространства листа существенно расширяет возможности, позволяя за одну операцию вычерчивания на бумаге выполнить несколько видов модели вместе с дополнительными графическими элементами чертежа, такими, как рамка листа, штамп, примечания и т.д.; управлять видимостью слоев, заданной по видовым моделям. Каждый такой вид может быть задан в собственном масштабе и с любой точки в 3-мерном пространстве и может содержать поперечный разрез, ортогональную проекцию или перспективное изображение. Любое изменение модели автоматически отражается на любом виде, связанным с ней.

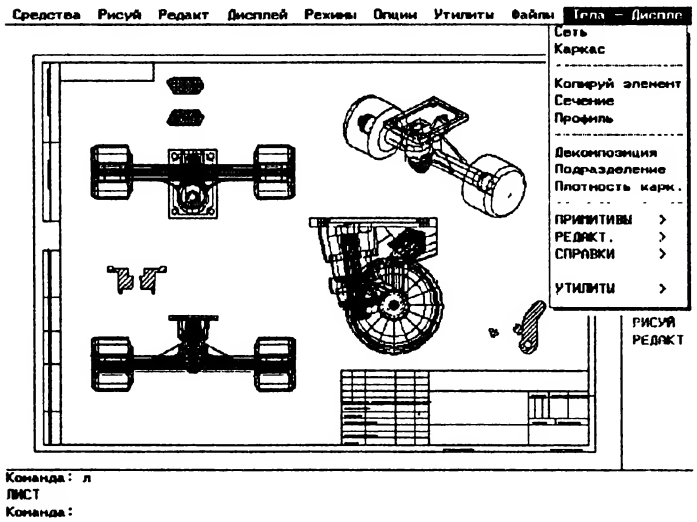


Рис. 1. Модель листа в Автокаде версии 11

Поддержка локальной вычислительной сети

Для пользователей, работающих с Автокадом в локальной сети, предусмотрены средства создания ссылок и блокировки файлов. Спроектированное для работы со многими типами сетей, это средство позволяет пользователю управлять графическими библиотеками.

Для сетевой поддержки предусмотрены:

- новая система блокировки файлов, предохраняющая от более одного одновременного доступа к файлу и исключающая тем самым потерю данных и сбой в работе Автокада;
- доступ к серверу сети, что позволяет установить единственную копию Автокада на сервер сети.

Внешние ссылки

С помощью команды ССЫЛКА в рисунок можно динамически добавить внешние ссылки на другие рисунки. Это позволяет работать с другими рисунками Автокада без полного добавления их в текущий рисунок и без изменения их содержания.

В структуру ссылок текущего рисунка могут входить более 16 миллионов рисунков. Кроме того, рисунок, на который имеется ссылка, может содержать ссылку на другой рисунок, тем самым определяя вложенную структуру. Так как файлы, на которые имеются ссылки, реально не добавляются к текущему рисунку, при каждом входе в рисунок они подгружаются. Следовательно пользователь может быть уверенным, что работает с последней версией файлов, на которые сделаны ссылки. Механизм внешних ссылок позволяет организовать бригадный метод работы над проектом при использовании Автокада в локальной сети.

Расширение по объемному конструированию

Дополнительно поставляемое расширение по объемному конструированию (РОК) предоставляет возможность создавать твердые тела внутри Автокада и определять их физические и материальные свойства. Предусмотрены средства объединения, вычитания и нахождения разности объектов. Модель можно воспроизвести в виде тонированного изображения или изображения без скрытых линий (рис.2). Одним из достоинств объемных моделей является то, что их можно анализировать. Например, можно определить такие свойства твердотельных объектов, как центр тяжести, массу, площадь поверхности, моменты инерции и т.д. Кроме того, можно получить сечение тела плоскостью или линию пересечения поверхностей тел. Команды РОК делятся на 5 групп: (1) команды создания твердотельных примитивов, (2) команды редактирования твердотельных объектов, (3) команды работы с дисплеем, (4) команды получения справок и (5) утилиты.

Имеется возможность создавать следующие твердотельные примитивы: ящик (параллелепипед), конус, цилиндр, шар, тор, клин, а также создавать тела путем вращения и выдавливания плоских профилей с возможностью задания конусности. Кроме того, имеется воз-

можность преобразования в твердые тела некоторых обычных объектов Автокада.

К командам второй группы относятся команды выполнения логических (булевых) операций (пересечение, вычитание, объединение), сопряжения скруглением и фаской, редактирования ГТК дерева с возможностью изменения примитивов, изменения материалов, разделения тел, создания поперечных сечений и копирования элементов.

Команды третьей группы позволяют получать список твердого тела, вычислять массо-инерционные характеристики, включая массу, объем, описывающий параллелепипед, моменты инерции и их производные, главные моменты и оси, площадь тела и др.

Команды четвертой группы позволяют выбирать тип отображения твердотельных объектов - каркасный или 2-мерную поверхностную сеть.

К командам пятой группы относятся команды перемещения тел, ПСК твердых тел, экспорт и импорт файлов .ASM системы AutoSolid, работа с файлами библиотек материалов, генерация профилей.

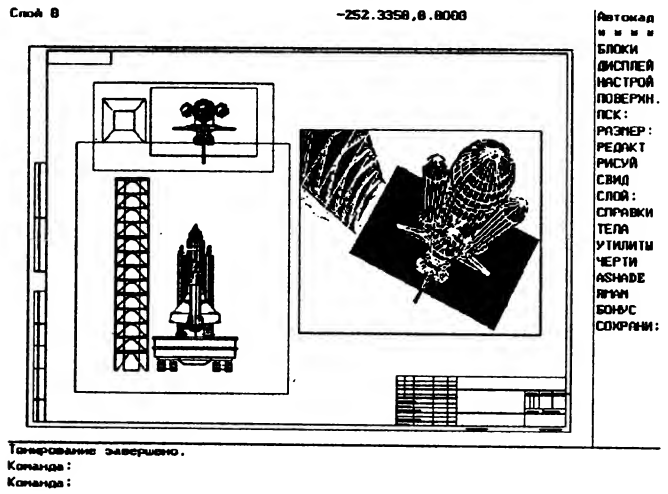


Рис. 2. Отображение моделей

Система разработки приложений Автокада и Автолисп

В дополнение к Автолиспу Автокад версии 11 обладает новой мощной системой разработки приложений Автокада. Эта система является средой для разработки программ на языке Си с их последующим выполнением в среде Автокада. Как и в случае Автолиспа, пользователь может определять новые команды Автокада, которые впоследствии могут быть вызваны вводом их имени из командной строки. Как и стандартные библиотеки Си, среда СРП определяется библиотеками и файлами прототипов. Для компиляции СРП программ можно использовать такие трансляторы языка Си, как High C, Watcom C и Zortech C, имеющие возможность компиляции программ в защищенном режиме. В следующей версии Автокада планируется предусмотреть возможность использования трансляторов в реальном режиме, например Microsoft C версии 6.0.

Приложения СРП, как правило, более эффективны по скорости и использованию памяти; кроме того, они в отличие от Автолиспа могут непосредственно обращаться к операционной системе и аппаратному обеспечению.

В Автолисп добавлены следующие функции:

- (ENTMAKE) позволяет добавлять примитивы и определения блоков в рисунок без использования команд, а также удалять существующие определения блоков;
- (NENTSEL) позволяет выбирать вложенные в определения блоков примитивы, отдельные вершины полилиний для различных приложений кроме примитивов, выбираемых функцией (ENTSEL);
- (WCMATCH) сравнивает строки со спецификациями строк глобальных символов и возвращает состояния соответствия;
- (CVUNITS) и соответствующий файл acad.unt позволяют преобразовывать различные единицы измерений;
- (XLOAD), (XUNLOAD) и (ADS) управляют загрузкой, выгрузкой и перечнем программ СРП;

В комплект поставки добавлены несколько новых Лисп-программ, а большинство существовавших Лисп-программ обновлено.

Расширенные данные примитивов

Добавлен механизм расширенных данных примитивов, позволяющий добавлять примитивам любые данные через DXF, Автолисп и СРП и группировать данные по именам "зарегистрированных приложений".

Восстановление испорченного рисунка

Утилита восстановления испорченного рисунка Autodesk - это по существу новый стандарт в области защиты данных. Новый пункт главного меню Автокада "восстановление испорченного рисунка" позволяет восстанавливать испорченные рисунки, а команда ПРОВЕРЬ - определять наличие ошибок в рисунке и восстанавливать его, не выходя из графического редактора Автокада.

Размеры

Для улучшения простановки размеров предусмотрены:

- символьная таблица размерных стилей, позволяющая сохранять, восстанавливать и модифицировать установки размерных переменных для отдельных размерных примитивов по имени размерного стиля;
- ординатные размеры;
- перемещение и редактирование размерного текста;
- задание для линейных размеров зазора между текстом и размерной линией и стрелками внутри или снаружи выносных линий;
- более гибкое размещение текста, стрелок и размерных линий для радиусов и диаметров;

- задание угловых размеров (так же, как и линейных) по трем точкам;
- задание высоты текста допуска относительно высоты основного размерного текста;
- наклонные размеры.

Пользовательский интерфейс:

- во все текстовые поля диалоговых окон добавлена возможность редактирования строк;
- строки текста могут быть длиннее полей диалоговых окон, для их просмотра применяется горизонтальная прокрутка;
- предусмотрено использование шкал прокрутки для просмотра длинных списков;
- предусмотрен вывод в диалоговых окнах, предназначенных для отображения списков, одновременно от 10 до 15 строк, а не 5 строк, как в прежних версиях;
- предусмотрены нажатие клавиши Enter и комбинации клавиш Ctrl+C вместо указания на кнопки "ДА" и "Отмена";
- существует возможность использования дилоговых окон файлов и их спецификаций при всех запросах имен файлов в графическом редакторе Автокада;
- имеется команда ДИАЛРЕД, позволяющая редактировать текст и атрибуты с помощью строкового текстового редактора в диалоговом окне.

Прочие изменения:

- добавлены две новые переменные среды ACADPLCMD и ACADPPCMD вместе с именем файла чертежа по умолчанию "AUTOSPOOL", что позволяет командам Автокада ЧЕРТИ и ПЕЧАТАЙ использовать буферные программы вывода на бумагу;
- в процессе установки системы предусмотрен этап ввода данных о пользователе, позволяющий добавить в исполняемую программу две ASCII строки, которые выводятся в заголовке главного меню (и других) при запуске Автокада. Запросами строк являются "Лицензиат:" и "Получено от:". После окончания установки данные строк изменить нельзя;
- переменной среды ACAD можно задавать несколько каталогов для определения пути поиска библиотек и вспомогательных файлов;
- добавлены восемь новых типов выключки текста, и поддерживаются любые комбинации "Верх/Середина/Базовая/Низ" и "Левая/Центр/Правая";
- точки можно вводить в трехмерных сферических или цилиндрических координатах;
- значения углов можно вводить в радианах и градах независимо от текущей установки угловых единиц с помощью соответствующего суффикса;
- большинство системных переменных можно изменить, просто введя их имя, т.е. нет необходимости сначала вводить имя команды УСТПЕРЕМ;
- при настройке плоттеров и принтеров можно выбрать параллельные порты ввода-вывода;
- примитивам, входящим в определения блоков, автоматически присваиваются (и поддерживаются) метки;
- добавлена команда ТЕНЬ для получения тонированных изображений на 256-цветных мониторах и для отображения без скрытых линий на мониторах с количеством цветов 16 и меньше;
- имеются две новые установки оптимизации вывода на плоттер, определяющие, должны ли совпадающие векторы удаляться или оставаться при вычерчивании;
- возможность поворачивать чертежи на 180 и 270 градусов дополнительно к имеющемуся ранее повороту на 0 и 90 градусов. Можно также поворачивать и трехмерные (неплоские) виды;
- добавлен новый примитив - произвольная сеть с соответствующей командой ПГРАНЬ, позволяющей создавать произвольную комбинацию граней с общими вершинами. Это дает возможность хранить такие грани в более компактной форме, чем такой же набор 3-мерных граней;
- добавлена поддержка 8-битовых и расширенных 16-битовых шрифтов для национальных версий Автокада;
- обновленный интерфейс с периферийными устройствами версии 4.1 поддерживает оптимальное использование драйверов мониторов при работе Автокада в пространстве листа.

Кроме того, были сделаны и некоторые другие улучшения и исправлены имевшиеся ошибки.

Имеются версии Автокада для широкого ряда персональных компьютеров и рабочих станций. Файлы рисунков Автокада полностью переносимы на компьютеры других типов без необходимости любых преобразований или трансляций. К числу поддерживаемых Автокадом платформ относятся:

- стандартные персональные компьютеры типа 286, 386 или 486 (то есть IBM AT и PS/2), работающие под управлением одной из операционных систем - MS/PC-DOS, OS/2, SCO Xenix 386 (специальная версия Автокада для компьютеров типа 386 и 486 позволяет пользователям иметь доступ к дополнительным возможностям среды встроенного в Автокад расширителя DOS) ;
- рабочие станции Sun Microsystems;
- рабочие станции DEC (под ULTRIX или VMS);
- персональные компьютеры Apple Macintosh II;
- рабочие станции Apollo;
- рабочие станции IBM RS 6000.

Справки по поводу деталей и системных требований Автокада для каждой платформы можно получить у полномочных дилеров Autodesk RF.

Рисунки, созданные всеми предыдущими версиями Автокада, могут использоваться в Автокаде версии 11 без каких-либо преобразований.

Поскольку во многих проектных организациях нашей страны используются компьютеры 286, следует особо отметить отличия Автокада 286 версии 11 от Автокада 386. В нем отсутствуют:

- команда ТЕНЬ;
- система разработки приложений Автокада и расширения по объемному конструированию.

ПРОБИТЬСЯ НА ЗАПАД

Фирма **Spirit**, Занимающаяся Экспортом Know-How,
Приглашает Разработчиков Оригинальных Программных Продуктов к Сотрудничеству

Рассматриваются любые завершенные программы, основанные на собственной технологии. Для рассмотрения Вашего проекта высылайте подробное описание, демонстрационную версию и квитанцию о денежном переводе (435 рублей для частных лиц и 2450 рублей для организаций) по адресу: *Москва, 117454, а/я 63. С пометкой "Трансфер".*

Наш счет: р/с 0467072 в КБ "ИнвестБизнесБанк" корр.счет 1161572 МФО 201791 в ГУ ЦБ РФ.
Тел. (095) 457 9216, 138 8781. Факс (095) 126 3156

Software **P**roducts **I**ntegrated **R**esearch **I**ntelligence **T**ransfer

Требования Автокада 386 к аппаратному обеспечению:

- COMPAQ Deskpro(R) 386, 386SX, IBM(R) PS/2(R) модели 70, 80, Hewlett-Packard 386 или полностью 386-совместимая машина. Процессор 486 также поддерживается как 80386-совместимый;
- математический сопроцессор 80287 или 80387;
- 2 Мбайта оперативной памяти. Первый мегабайт содержит 640 Кбайтов обычной памяти. Второй мегабайт памяти должен быть доступен как расширенная память типа extended исключительно для Автокада 386. Рекомендуемый размер раширенной памяти - 4 мегабайта;
- жесткий диск, как минимум, с 20 Мбайтами свободного пространства. Минимальная установка, состоящая из исполняемых и вспомогательных файлов, требует около 2 Мбайтов. Полная установка требует 8.7 Мбайтов на диске.

Автокад для Windows является расширением к обычному Автокаду для среды DOS. Он поддерживает все функции, имеющиеся в Автокаде для DOS, плюс несколько дополнительных возможностей, таких как динамический обмен данными и современный графический пользовательский интерфейс, соответствующий стандартам среды Windows, например, программируемые адаптируемые инструментальные строки меню (tool bar) и диалоговые окна.

В Автокаде для Windows имеется возможность запуска пакетных файлов команд при работе программы в фоновом режиме и режиме автоматического обновления данных при ее восстановлении. Имеется возможность выполнять обратную прокрутку текстового окна Автокада для просмотра выполненных ранее команд.

Двухсторонний динамический обмен данными (DDE) позволяет связать Автокад, например с таким пакетом, как Microsoft Excel - можно установить динамическую связь между электронными таблицами Excel и рисунком в графическом редакторе Автокада - любое изменение данных в электронных таблицах тут же будет отражаться в рисунке и наоборот редактирование рисунка Автокада будет автоматически изменять информацию в соответсвующей базе данных в Excel, содержащую, например спецификацию материалов.

Для передачи всего рисунка или его фрагмента из Автокада в другое приложение для Windows, например в MS Word for Windows или в PageMaker, можно использовать утилиту Clipboard системы Windows, что очень удобно для применения в системах типа настольной типографии (DeskTop Publishing).

СРП Автокада для Windows позволяет использовать такие широко распространенные трансляторы как Microsoft Wisual Basic, Microsoft Quick C for Windows, Microsoft C версии 6.0 и Borland C++ версии 3.0.

Для новичков в Автокаде в комплект Автокада для Windows входит электронное справочное руководство, занимающее около 17 Мбайтов дискового пространства, с иллюстрациями, пояснениями команд и перекрестными ссылками, и файл программы справочной информации с возможностью получения контекстной справки и использования гипертекста.

Системные требования Автокада для Windows аналогичны требованиям Автокада версии 11(386). Быстродействие Автокада для Windows меньше, чем у обычного Автокада, но фирма Autodesk связывает улучшение быстродействия с появлением на рынке 32-разрядной версии системы Windows. В будущем планируется создание 32-разрядной версии Автокада для среды Windows NT.

К основным особенностям Автокада версии 12, о которых уже сейчас можно с уверенностью говорить, относятся программируемые диалоговые окна и отсутствие главного меню. Следует заметить, что данная версия уже не будет работать на компьютерах 286: минимальным требованием является компьютер типа 386 с несколькими мегабайтами расширенной памяти.

Автокад и другие программные продукты фирмы Autodesk продаются через сеть квалифицированных полномочных дилеров, в функции которых входят - продажа и техническое сопровождение программ, консультации конечных пользователей и обмен на новые версии программ. Кроме того, имеется и развивается сеть авторизованных учебных центров, проводящих обучение работе с системой Автокад по различным курсам, а также сеть авторизованных разработчиков, создающих свои собственные прикладные системы на базе Автокада. Регулярно выпускаются каталоги таких приложений для Автокада. ■

Анализ отечественных систем защиты программ от копирования

Часть 2. Война форматов

В. Вегнер, А. Крутяков,
В. Серегин, В. Сидоров, А. Спесивцев

Введение

Можно сказать, что сейчас на рынке программных средств идет настоящая "война форматов". Объясняется она тем, что наиболее простой способ взлома хэкером систем защиты состоит в копировании ключевой дискеты. Поэтому способность системы противостоять взлому определяется в первую очередь именно стойкостью изготавливаемых для нее ключевых дискет. В этой части статьи мы проведем анализ форматов ключевых дискет ряда отечественных систем защиты от копирования, и адресуется она прежде всего читателю достаточно подготовленному. Предвидя возможные возражения относительно публикации форматов ключевых дискет, мы хотим сказать следующее. Сама по себе информация о формате ключевой дискеты не делает систему менее защищенной. Действительно для неспециалиста это - китайская грамота. Необходимо еще очень много узнать, чтобы воспользоваться данной информацией для изготовления копии ключевой дискеты. Сделать это может только специалист. Однако если уж за дело взялся профессионал, то он всегда сможет получить необходимую информацию с помощью широко распространенных программ (типа DISK EXPLORER).

Форматы ключевых дискет

Этот раздел написан на основе анализа ключевых дискет, изготавливаемых для ряда отечественных систем. По нашему мнению, данная информация может оказаться полезной для понимания основных принципов изготовления дискет, не копируемых стандартными или некоторыми специализированными средствами. Вместе с тем данные форматы достаточно просто копируются с помощью платы COPY II PC OPTION BOARD DELUXE (в дальнейшем Option Board), и поэтому не могут служить действительно надежной основой при проектировании систем. Вся процедура копирования ключевой дискеты с помощью платы занимает не более 10 минут. Разработка форматов, не копируемых с помощью платы, является достаточно сложным делом, требующим как понимания работы контроллера НГМД, так и самой платы. Однако только такие форматы действительно надежно защищают ключевую дискету от копирования.

JAWS

Ключевая информация в системе располагается на дорожке 0 инженерного цилиндра с номером 42. Автор системы защиты от копирования JAWS утверждает, что созданные дискеты не копируются программой COPYWRIT, что не соответствует действительности. Дискета копируется программой COPYWRIT, если задать максимальный номер копируемого цилиндра, равный 42. Естественно дискета легко копируется и с помощью платы Option Board, если указать копирование до цилиндра 42 включительно.

Формат ключевой информации

Дорожка 0 цилиндра 42 форматируется на 9 секторов с обычными параметрами (GAP=80, N=2). При этом применяется метод нарушения последовательности секторов. Первый сектор на дорожке имеет номер 11, остальные - последовательно от 2 до 9.

Следует отметить, что программа проверяет только наличие сектора с номером 11. Поэтому достаточно иметь на ключевой дорожке обычный сектор с номером 11, а остальные не нужны.

HOTA

Авторы этой системы также решили расположить информацию о защите в области инженерных цилиндров. В отличие от системы JAWS HOTA использует дорожку 0 цилиндра 41. Однако программа COPYWRIT и плата OPTION BOARD также успешно копируют эту дискету.

В системе применен более любопытный по сравнению с JAWS формат ключевой дорожки.

Используется один из вариантов метода нарушения значений CHRN, который заключается в следующем. Ключевая дорожка форматируется на 10 секторов с параметрами GAP=16, N=2. В поле N всех секторов записываются значения 3. Перезаписывается сектор с номером 1. Это приводит к затиранию маркера идентификатора следующего сектора. На дорожке остаются 9 секторов. Значения CHRN затираемого сектора не играют существенной роли. Номера оставшихся секторов нумеруются обычным образом от 1 до 9. В сектор с номером 1 записывается ключевая информация.

Как и в системе JAWS, проверяется только наличие сектора с номером 1 (R=1) и N=3 и из него считывается информация. Поэтому на ключевой дорожке достаточно иметь только этот сектор и записанную в него информацию. Остальные секторы не проверяются. Копирование проще всего сделать, отформатировав ключевую дорожку дискеты-копии на 1-й сектор с параметром N=3 и записав в него информацию из первого сектора ключевой дорожки оригинальной дискеты.

SHIELD

Авторы системы SHIELD не пошли проторенным путем - отказались от инженерных цилиндров и нарушений CHRN. Они использовали свободное пространство на дорожке 0 цилиндров 0, 1 и 2 для размещения ключевой информации. Дискета не копируется напрямую программой COPYWRIT, однако после такого копирования достаточно немножко

помочь программой DISK EXPLORER, и вы будете иметь работоспособную копию. Естественно дискета легко копируется платой OPTION BOARD.

На дискете применяется метод форматирования с кодом длины 1. Рассмотрим формат ключевых дорожек.

Все ключевые дорожки форматируются на 19 секторов с N=1 и GAP3 между 3 и 7. В данном случае длина GAP3 не имеет значения, так как она не проверяется. В нечетных секторах в CHRN устанавливается N=2 и номера секторов R принимают последовательные значения от 1 до 9. Перезапись этих секторов затирает адресные маркеры идентификаторов примыкающих секторов, и получается стандартный формат с N=2 и номерами секторов R от 1 до 9. Последний сектор используется для контроля ключевой информации.

На цилиндре 0 последний сектор имеет параметры N=6 и R=11 и используется для контроля длины дорожки. При чтении этого сектора можно получить длину GAP4B и увидеть начало дорожки.

На цилиндрах 1 и 2 последний сектор имеет параметры N=1 и R=10. В эти секторы записывается ключевая информация.

Для получения работоспособной копии дискеты после копирования программой COPYWRIT можно воспользоваться программой DISK EXPLORER. Нужно только скорректировать последний сектор цилиндра 0, чтобы программа проверки ключевой дискеты получала необходимые данные при чтении этого сектора, для чего необходимо прочитать ключевой сектор с оригинальной дискеты и выполнить короткую запись его на дискету-копию. Длина записи должна быть достаточной для перезаписи GAP4B и всех данных начала дорожки. Чтобы при короткой записи не затереть маркер идентификатора первого сектора на дорожке, рекомендуем при форматировании ключевой дорожки на цилиндре 0 использовать маленький GAP порядка 3. Это обеспечит достаточно большую длину GAP4B и необходимое место для записи данных, контролируемых программой. Не забудьте переписать содержимое других секторов на этих дорожках, так как в них содержится информация для работы DOS.

PROTECT – INST_FD

Авторы этой системы также не удержались от примечательной идеи использования инженерных цилиндров. Нестандартный формат применяется на дорожке 0 цилиндров 0 и 41. Дискета не копируется программами COPYWRIT и COPYIIPC, но не в силах устоять перед платой OPTION BOARD.

На дискете применяется метод форматирования с длиной 1, а также метод увеличения длины последнего сектора для запутывания программ COPYWRIT и COPYIIPC. Применяемый формат имеет следующие характеристики.

Формат одинаков для цилиндров 0 и 41. Выполняется форматирование на 11 секторах с N=1, GAP=255 и символом - заполнителем формата "X". Первые 9 секторов имеют стандартные R от 1 до 9 и N=2. Предпоследний CHRN имеет R=11 и N=6. У последнего сектора поля CHRN соответственно равны 123, 17, 249 и 7. Полученный формат при работе с секторами от 1 до 9 имеет "отрицательный" GAP3, так как форматирование выполняется с кодом длины 1, а операции с секторами выполняются с кодом длины 2. При этом CRC сектора залезает на SYNC адресного маркера идентификатора следующего сектора. В первом секторе цилиндра 41 записывается ключевая информация, а секторы 2, 3, ... , 9 заполняются символами F6h, что должно маскировать их под обычный формат.

Вместе с тем авторы проверяют из всего объема ключевой информации только информацию из первого сектора на цилиндре 41. Поэтому при копировании достаточно завести обычный сектор с номером 1 на цилиндре 41 и переписать туда ключевую информацию.

CONVOY, SOFTLOCK

Дискета не копируется программами COPYWRIT и COPYIIPC, но успешно копируется платой OPTION BOARD.

Все дорожки форматируются стандартным образом. Ключевая информация в системе располагается в поле GAP3 сразу за полем CRC первого сектора на цилиндре 0 дорожки 0.

Рассмотрим способ построения ключевой дискеты. Для ее изготовления может быть применен метод прекращения операции контроллера. Выполняется "короткое" форматирование на один сектор дорожки 0 цилиндра 0 с прекращением операции в GAP 2. Параметры CHRN сектора - C=0, H=0, R=1 с кодом длины сектора N=3. Такое форматирование полностью сохраняет данные на дискете, но позволяет перезаписывать данные сектора, включая GAP3. Затем выполняется "короткая" запись данных ключевого сектора, причем записываются поле данных сектора, поле CRC и данные GAP3. Прекращение операции выполняется в поле GAP3 после записи ключевой информации. Затем производится "короткое" форматирование на один сектор дорожки 0 цилиндра 0 с прекращением операции в GAP2. Параметры сектора CHRN следующие: C=0, H=0, R=1, при коде длины сектора N=2. Такое форматирование восстанавливает исходный формат дорожки.

SUPER GUARD

Система SUPER GUARD использует схему защиты, сходную с системами CONVOY и SOFT LOCK, но для хранения ключевой информации предназначается поле GAP3 первого сектора дорожки 0 цилиндра 1.

Помимо этого авторы системы для усиления защиты на дисководах, обеспечивающих 80 цилиндров (дисководы 1.2 Мбайта), записывают данные на цилиндр, не используемый при форматировании дискеты на 360 Кбайтов. Дорожка 0 цилиндра с абсолютным номером 31 (для накопителя, обеспечивающего 80 цилиндров) форматируется с параметрами GPL=84, N=1, H=0 на 9 секторов с номерами секторов R, соответственно равными 1, 2, 3, 244, 5, 6, 7, 10, 9. Ключевые данные записываются в сектор с номером R, равным 244. Такой метод позволяет защитить дискету от копирования платой OPTION BOARD при использовании на дисководах 1.2 Мбайта (на дисководах в 360 Кбайтов копия дискеты естественно также работает). Это объясняется тем, что плата OPTION BOARD не копирует промежуточные цилиндры для дискет емкостью 360 Кбайтов.

GUARD

Дискета не копируется программами COPYWRIT и COPYIIPC, но успешно копируется платой OPTION BOARD.

Авторы системы не отказали себе в удовольствии использовать инженерные цилиндры. Вместе с тем ключевая информация расположена практически на всей дискете.

Дорожки 0 цилиндров 0, 1 и 2 форматируются обычным образом на 9 секторов. Остальные дорожки всех цилиндров, включая цилиндр 41, форматируются на 5 секторов с параметрами GPL=192, кодом длины сектора N=3, символом - заполнителем формата 42 и номерами секторов R, соответственно равными 1, 2, 3, 4, 5. При этом данные записываются во все секторы дорожки 0 на всех цилиндрах. На дорожке 1 данные записываются в секторы только на цилиндрах 0, 1 и 2. Ключевые данные также записываются в сектор с номером 5 на дорожке 1 цилиндра 41. Все остальные секторы после форматирования не перезаписываются. Вполне возможно, что информация в эти секторы не записывалась, т. к. объем защищаемых данных был меньше.

STAS-3

Автор системы Стас Ляшев (см. систему HOTA) не отказался от идеи использования инженерных цилиндров. Ключевая информация располагается на поверхности 0 цилиндра 41. Дорожка форматировалась на 3 сектора с кодом длины сектора 2 (стандартный сектор 512 байтов), GAP=84 и символом - заполнителем 246. Значения CHRN секторов следующие - (41,0,11,2), (41,0,12,2) и (41,0,99,3). В последний сектор (с кодом длины 3) записана ключевая информация. Для получения работоспособной копии информацию из последнего сектора ключевой дискеты необходимо переписать в соответствующий сектор дискеты - копии.

ALFA – 2.05 (RAIZOR)

Система ALFA тоже использует крайне простой формат ключевой дискеты. Поверхность 0 цилиндра 39 отформатирована на 19 секторов с кодом длины сектора 1 (256 байтов), GAP=10 (точно не анализировали за ненадобностью) и символом - заполнителем 246. Значения CHRN первых 9 секторов: C=39, H=0, R=1-9 и N=1. Остальные 10 секторов имеют нулевые значения во всех полях CHRN. Ключевая информация записана в первом секторе. Вообще говоря, для копирования дискеты достаточно отформатировать соответствующую дорожку

дискеты-копии на 1 сектор с кодом длины 1 и CHRN (39,0,1,1) и перезаписать туда информацию из ключевой дискеты.

Дискета может быть скопирована с помощью программы DISK EXPLORER.

ECP

Авторы ECP похоже не пошли дальше использования средств BIOS при создании ключевой дискеты. Ключевая информация располагается на инженерных цилиндрах. Поверхность 0 цилиндра 40 форматировалась на 30 секторов с кодом длины сектора 0, GAP порядка 10 (этого достаточно) и символом - заполнителем 11. В первом, четвертом, седьмом и т. д. секторах в CHRN устанавливается N=2 и номера секторов R принимают последовательные значения от 1 до 9. Перезапись этих секторов затирает адресные маркеры идентификаторов двух примыкающих секторов, и получается стандартный формат с N=2 и номерами секторов R от 1 до 9. Сектор 28 не затирается и для него в CHRN при форматировании устанавливается равным (40,0,10,1). В этот сектор записывается ключевая информация. После записи затирается адресный маркер идентификатора одного примыкающего сектора. В результате на дорожке остается один последний сектор, используемый для контроля GAP4B, который имеет CHRN (40,0,11,2). Все это легко повторить с помощью программы DISK EXPLORER. Для сохранения длины GAP4B необходимо выполнить "короткую" запись информации из этого сектора с ключевой дискеты. О методе "короткой" записи рассказывалось при анализе системы SHIELD в предыдущем разделе.

Ключевая информация располагается также на поверхности 1 цилиндра 40. Там приводятся несущественные моменты со стандартными параметрами форматирования (количество секторов = 9, N=2 и символ - заполнитель 11). Вся игра происходит на длине GAP и CHRN секторов. Анализ структуры дорожки и форматирование легко сделать с помощью программы DISK EXPLORER. Никакая дополнительная информация на эту дорожку не записывается.

FILE_PROTECTION

Ключевая информация располагается на поверхности 0 инженерного цилиндра 40. Дорожка форматировалась на 11 секторов с кодом длины сектора 1 и GAP=232. В каждом случае используется свой символ - заполнитель. Способ его определения мы опишем ниже. При первичном форматировании CH секторов 40,0, а R и N следующим образом:

(1,2),(16,0),(17,0),(9,2),(5,3),(6,3),(7,3),(8,3),(9,3),(10,3), (11,3).

Затем производится запись в сектор с RN (11,3), что приводит к затиранию первого сектора дорожки, потом запись в сектор с RN (10,3) и т.д., включая и сектор с RN (9,2). В результате получается формат дорожки с 3 секторами с RN - (16,0),(17,0),(9,2). Теперь в сектор с RN (9,2) переписывается информация из соответствующего сектора ключевой дискеты.

На ключевой дискете с FILE_PROTECTION также был отформатирован инженерный цилиндр 40 поверхность 1 с CH (40,1) и остальным аналогичным форматом.

Разработчики системы не проверяют многое из того, что у них получается в формате дорожки. Поэтому для получения ключевой дискеты достаточно выполнить форматирование на 3 сектора с кодом длины сектора 2, RN (16,0),(17,0),(9,2), GAP=80 и перезаписать сектор (9,2) с ключевой дискеты.

Для определения символа - заполнителя достаточно прочитать сектор с RN (17,0) с ключевой дискеты и посмотреть код первого символа, который в нем записан. Это и есть символ - заполнитель формата.

Все перечисленные процедуры легко выполняются с помощью программы DISK EXPLORER.

ЗАКЛЮЧЕНИЕ

Результаты анализа перечисленных систем показывают однотипность используемых методов размещения ключевой информации в поле GAP3 (CONVOY, SOFT LOCK, SUPER GUARD). Это позволяет говорить о возможном заимствовании схемы построения ключевой дискеты. Впервые такая схема была применена в системе защиты кооператива "Эклипс".

Исследованию подвергался еще ряд систем, не включенных в обзор. Приведем их краткое описание.

Система, с помощью которой выполнялась защита русской версии пакета GEM фирмы DIGITAL RESEARCH, копируется достаточно легко.

Система защиты русификатора WINDOWS 3.0 (CYRWIN) не только легко копируется, но в ней очень просто изменить счетчик числа установок.

Авторы анализировали еще ряд различных систем, названий которых и параметров форматирования уже не помнят, т. к. не обнаружили там принципиально новых идей. ■

СП «ДИАЛОГ» & 2B Programmers Group предлагают средства защиты информации

CERBERUS
система защиты программ от копирования

- защищает все типы программ
- привязывает программу к дискете и компьютеру
- создает некопируемые ключевые дискеты
- не вскрывается с помощью отладчиков и эмуляторов
- включает средства Cerberus Kit, позволяющие защитить любые файлы (в том числе текстовые и баз данных)

КРИПТОН
аттестованная система защиты информации любого грифа секретности

- разграничивает доступ к логическим дискам
- опознает пользователя по двум ключевым элементам (паролю и ключ-дискете)
- блокирует загрузку компьютера с дискеты
- осуществляет прозрачное шифрование данных

разработана в содружестве с фирмой АНКАД

КРИПТОША
система защиты конфиденциальной информации

менее дорогая, более быстрая, но не предназначенная для борьбы с ЦРУ, версия системы КРИПТОН

разработана в содружестве с фирмой АНКАД

Адрес: 115598, Москва, ул. Ягодная, 17, СП «ДИАЛОГ»
телефон: (095) 329-4611
E-mail(RELCOM): QA-DEP@JVD. MSK.SU

Зарубежная компьютерная хроника

1992 год, февраль

Японская корпорация NEC заявила, что, начиная с середины февраля, она будет экспортировать в Соединенные Штаты notebook-компьютеры с цветными дисплеями на жидких кристаллах. NEC надеется продавать 3.000 ПК за месяц и рассчитывает на 5% рынка США.

Токио - Корпорация Intel, крупнейшая американская компания по производству полупроводников, заключила соглашение с японской компанией Sharp - производителем потребительской электроники о совместном производстве чипов флэш-памяти. Intel и Sharp договорились объединить усилия разработчиков, с тем чтобы Sharp могла приступить к производству флэш-кристаллов в 1994 году. Аналитики предсказывают рост спроса на флэш-память во всем мире до 1.5 млрд. долл. к 1995 году (по некоторым оценкам до 10 млрд. долл. в течение 10 лет). Сейчас компания Intel владеет 85% мирового рынка кристаллов. Ее конкурентами, ведущими собственные разработки, являются корпорации NEC, Toshiba, Mitsubishi, Fujitsu.

Компании AT&T (США) и "Дальняя связь" (Россия) образовали совместное предприятие для адаптации и торговли оборудованием телефонных сетей. Начальные капиталовложения составили несколько миллионов долларов. Соглашение, подписанное 12 февраля, предусматривает, что компания AT&T Network Systems International будет владеть 68% предприятия, названного AT&T St. Peterburg. AT&T сообщила, что вторая часть СП будет принадлежать русскому участнику - компании "Дальняя связь".

Digital Equipment Corp. представила микропроцессор Alpha: 64-разрядный, работающий с тактовой частотой 200 МГц и имеющий пик производительности 400 MIPS; производительность при повседневной эксплуатации будет сохраняться на уровне 150 MIPS. Машины на основе Alpha будут работать под операционными системами WINDOWS NT и VMS.

Продажи персональных компьютеров и периферийного оборудования японского производства в финансовом году, заканчивающемся 31 марта, упали на 7-10% по сравнению с предшествующим годом, давшим около 1.1 триллиона иен (8.6 млрд. долл.), сообщила Ассоциация электронной промышленности Японии (the Electronic Industries Association). По сообщению представителей Ассоциации, общий объем производства японских персональных компьютеров, как полагают, будет в итоге между 2.2 и 2.3 млн. устройств, снижение составит около 15%. Представители Ассоциации ссылаются на резкое падение спроса на компьютеры во всем мире и замедление темпов роста японской экономики как причины печальной ситуации в промышленности.

NEC Corp. сообщила, что она с двумя телекоммуникационными фирмами Дании получила заказ от компании из России на создание цифровой телекоммуникационной сети, предназначенной для соединения России и Дании. Контракт подписали Российская телекоммуникационная компания Sov Telecom ("Телеком"), NEC, Great Northern Telegraph Co. и Telecom Denmark Ltd. Они будут работать вместе над созданием оптической кабельной сети между двумя странами до конца этого года.

1992 год, март

Microsoft Corp. сообщила, что она приобретет Fox Software Inc. в обмен на акции стоимостью в 173.4 млн. долл. Предполагаемая сделка, крупнейшая из когда-либо предпринимавшихся Microsoft, позволит Microsoft выйти на рынок программного обеспечения баз данных для персональных компьютеров, оцениваемый в 1 млрд. долл. Это единственный из основных рынков программного обеспечения, на котором компания пока не появилась. Совершив это приобретение, фирма Microsoft должна начать борьбу за раздел рынка с Borland International Inc., лидером в производстве программного обеспечения для баз данных.

В апреле начинается продажа двух конкурирующих операционных систем: OS/2, производства IBM, и Windows 3.1, производства Microsoft. Фирма IBM обратилась ко всем своим служащим с просьбой предлагать и продавать OS/2, предусмотрев различные виды стимулов. Аналитики считают, что в течение двух лет Windows будет продаваться дешевле, что обусловлено величиной торговых и накладных расходов фирмы IBM, а также необходимостью платить компании Microsoft 20 долл. с каждой копии OS/2 в качестве лицензионного вознаграждения за использованное программное обеспечение.

Соединенные Штаты 30 марта предупредили японское правительство о своем "серьезном беспокойстве", вызванном "застоем" доли иностранных производителей на рынке полупроводников и советовали японским компаниям увеличить их закупки кристаллов по импорту.

Предупреждение , сделанное миссис Карлой Хиллс (Carla Hills), торговым представителем США, явилось следствием подсчитанной сообща статистики, показывающей, что иностранная доля повысилась только незначительно от 14.3% до 14.4% в последнем квартале прошлого года.

1992 год, апрель

Консультативный совет по надежности и безопасности компьютерных систем при правительстве США отложил принятие стандарта по контролю электронных сообщений до июня 1993 года. Проект стандарта цифровых сигнатур был предложен в прошлом году Национальным институтом стандартов и технологии Министерства торговли США. Отсутствие стандарта, позволяющего однозначно идентифицировать пользователя компьютера, затрудняет, например, оплату счетов и подписание контрактов. Необходимо принятие правительственного плана по вопросам шифрования и кодирования.

По материалам газет The Wall Street Journal Europe и The Guardian.

Индикатор заражения программы компьютерным вирусом

В.Водолазкий

Вот уже несколько лет нас пугают всепожирающей "компьютерной чумой" - эпидемией компьютерных вирусов. Но все средства защиты, как правило, ограничиваются применением одной из версий AIDSTEST или -V (иногда довольно старой) и какого-нибудь резидентного фильтра типа FLUSHOT или ANTI4US2. Созданием этих программ занимается небольшая кучка энтузиастов, а основная масса пользователей и программистов занимает выжидательную позицию - кто кого. В самом деле, чем-то ведь эта борьба должна кончиться: или с вирусами будет покончено раз и навсегда, или будет поставлен крест на использовании ЭВМ. Так что, поживем, увидим...

Но ведь с точки зрения здравого смысла, разработчик программного обеспечения должен нести полную ответственность за свое произведение. И если программа не в состоянии сама восстановить себя в случае ее инфицирования, то по крайней мере она должна сигнализировать пользователю о факте заражения. Это означает, что при разработке прикладной программы необходимо включить в нее средства, позволяющие обнаружить появление в ней чужеродного кода.

Рассмотрим основные пути решения этой задачи. Обнаружение вирусной атаки, вообще говоря, может быть осуществлено путем сравнения некоторых характеристик исполняемой программы с их эталонными значениями. Примерами таких характеристик могут служить : длина программы, ее контрольная сумма, окрестности точки входа в программу и даже стиль программирования. Но поскольку данная процедура носит вспомогательный характер и не призвана полностью заменить специализированные антивирусные средства, используемые в ней признаки инфицирования программы должны вычисляться с минимальными затратами вычислительных ресурсов и в то же время достаточно надежно. В наибольшей степени этим противоречивым условиям удовлетворяет проверка длины выполняемого файла.

Казалось бы, что проще? Открыть файл, переместить указатель в конец (функция DOS 4202), и считывай длину файла из регистровой пары DX:AX. Но вопрос в том, какой файл? Если ввести в программу проверку длины файла с фиксированным именем, например SUPERPRO.EXE, то где гарантия, что пользователь, изнуренный необходимостью вводить такое длинное и неудобное название, не переименует программу, например в SP.EXE, или просто не попытается вызвать его из другого текущего каталога?

К счастью, DOS сохраняет информацию о полном (включая имя устройства и путь доступа) имени исполняемой программы. Необходимо только воспользоваться этой информацией. Для чего мы должны получить доступ к окружению DOS. Окружение DOS представляет собой последовательность ASCIIZ строк (строк в коде ASCII, завершающихся нулевым байтом), в которых хранятся так называемые переменные среды. Кроме того, в окружение, начиная с версии DOS 3.0, включено и полное имя исполняемой программы. На рисунке приведен типовой состав окружения, записанный в мнемонике ассемблера. Обратите внимание, что имя загружаемой программы в отличие от переменных среды выделено словом $0001. Для доступа к окружению DOS вам необходимо получить сегментный адрес PSP текущей программы (функция DOS $62), а затем считать адрес окружения из PSP по смещению $2C. Пример реализации описанного подхода приведен в листинге 1.

```
db 'COMSPEC=C:\4DOS.COM',0
db 'PROMPT=$p$g',0
db
'PATH=E:\WIN3;C:\DOS;D:\4DOS;C:\TASM20;C:\TDE
BUG;E:\TP;D:\ME',0
db 'TEMP=e:\WIN3\TEMP',0
db 0
dw 1
db 'C:\hlup\bimbom\superpro.exe',0
db 0
```

Рисунок. Представление окружения DOS в мнемонике ассемблера

"На Аллаха надейся, а верблюда привязывай."
Синдбад-мореход.

С целью экономии объема памяти процедура определения длины файла реализована на ассемблере. Приведенная в листинге 1 программа ориентирована на использование компилятора Turbo Assembler 2.0 фирмы Borland, поддерживающего модель данных Turbo Pascal. Необходимая интерфейсная часть и оформление индикации на экране дисплея показаны в листинге 2. Обратите внимание на управляющую директиву {$F+}. Дело в том, что при использовании модели TPASCAL в программе на ассемблере автоматически генерируются процедуры типа FAR (дальний межсегментный вызов и возврат). Сам же Турбо Паскаль вызовы оптимизирует, и если указанная директива будет отсутствовать, это неизбежно приведет к зависанию программы.

Установка процедуры в программу осуществляется следующим образом. Вначале вы отлаживаете программу без включения модуля VIRCHECK. Когда программа полностью отлажена, добавьте к списку используемых модулей в директиве USES имя модуля VIRCHECK. Первым оператором в основной процедуре программы сделайте вызов:

```
SelfTest(0);
```

При запуске программы процедура обнаружит несоответствие эталонной длины программы (0) фактическому значению, которое будет выдано на дисплей. После этого вы вновь входите в редактор и заменяете аргумент процедуры SelfTest на значение, выданное системой. В примере 1 приведена простая программа "Нарцисс", которая любуется собой и проверяет свою собственную длину.

```
program test;
uses Vircheck;
begin
    SelfTest(7502);
end.
```

Пример 1. Обращение к процедуре проверки длины загружаемого файла

Предложенные в данной статье решения ориентированы на пользователей Турбо Паскаль 5.0 и 5.5. Программисты, работающие на Турбо Паскале 6.0, в более выгодном положении - они могут использовать встроенный ассемблер, поместив текст, приведенный в листинге 1, в конструкцию, представленную в примере 2.

```
Procedure CalcFile( var LL : LongInt);
begin
   asm
.....     текст процедуры из листинга 1
.....
   end
end;
```

Пример 2. Использование встроенного ассемблера Турбо Паскаль 6.0

Необходимо только исключить оператор RET из текста на ассемблере, так как он будет добавлен автоматически при компиляции программы.

Если вас очень пугает "огромная" длина модуля (с текстовым оформлением - около 1 Кбайта), вы можете сделать его оверлейным. И еще одно последнее замечание. Если вы используете пакет PKLITE для динамического архивирования исполняемого файла, вам придется устанавливать в SelfTest длину файла ПОСЛЕ его архивирования.

Господа, не забывайте, что успех и признание пользователей обеспечены только НАДЕЖНОМУ программному обеспечению. ■

Листинг 1.

```
        NAME    VIRCHECK
        PAGE    58,120
                ;
        .286C           ; только для программ, предназначенных
                        ; для эксплуатации с 80286
        .RADIX  16      ;устанавливаем по умолчанию 16-ную систему
                        ;счисления
;
;       В.В.Водолазский, (С) "Монитор" 1991
;
;       Процедура определения длины
;       текущего исполняемого файла
;
;       Входит в состав модуля VirCheck
;       совместима с Turbo Pascal 5.0+
;
;       Для получения объектного файла
;       используйте командную строку:
;
;       TASM VIRCHECK
;
;
MODEL TPASCAL
LOCALS
CODESEG
;
; Вызов процедуры из программы на TurboPascal
; Procedure CalcFile(var L : LongInt);
    PUBLIC  CALCFILE
PROC    CALCFILE L : DWORD
    PUSHF
    CLD                         ;сброс флага направления
    MOV     AH,62
    INT     21                  ;читаем адрес активного PSP
    MOV     ES,BX
    MOV     BX,2CH              ;es:bx - адрес среды
    MOV     AX,WORD PTR ES:[BX]
    MOV     ES,AX
    XOR     DI,DI               ;теперь пара ES:DI настроена на
                                ;поиск в переменных среды
    MOV     CX,8000             ;размер среды в байтах
    MOV     AL,1                ;это слово является индикатором
                                ;имени исполняемого файла
    REPNE   SCASB               ;ищем байт $01
    INC     DI                  ;коррекция указателя
    ; теперь ES:DI указывают на начало ASCIIZ-строки
    ; с полным именем (например, C:\hlup\bimbom.exe),
    ; исполняемого в данный момент файла
    PUSH    DS                  ; определение DS глобально для прог-
                                ; раммы и должно быть сохранено
    PUSH    ES
    POP     DS
    PUSH    DI
    POP     DX                  ; DS:DX := ES:DI
    MOV     AX,3D00             ; открываем файл
    INT     21
    JNC     @@1                 ;если не было ошибки
    MOV     AX,4CF1             ;если файл открыть не удалось
    INT     21                  ;выход из программы
@1: MOV     BX,AX               ;сохраняем номер канала в/в
    MOV     AX,4202             ;перемещаем указатель в конец файла
    XOR     CX,CX
    MOV     DX,CX               ; В DX:AX возвращается длина файла
    INT     21                  ; (т.е. максимальное смещение указа-
                                ; теля)
    PUSH    AX
    MOV     AH,3EH              ;закрываем файл
    INT     21
    POP     AX
    POP     DS
    LES     DI,[L]              ; передаем результат измерений в
    STOSW                       ; переменную связи
    MOV     AX,DX
    STOSW   POPF
    RET
ENDP
END
```

Листинг 2.

```
unit VirCheck;
{ В.В.Водолазский (С) "Монитор" 1991 г.
Модуль проверки длины текущего исполняемого
файла. Для работы требует наличия объектного
файла VIRCHECK.OBJ.
Совместим с TurboPascal 5.0+
}
{$F+} {$O+}
INTERFACE
    uses CRT;
    Procedure SelfTest(Len : LongInt);
IMPLEMENTATION
{$L vircheck.obj}
    Procedure CalcFile(var L: LongInt); external;
    Procedure Buzzer;
    var i : byte;
    BEGIN
        for i := 1 to 5 do
            begin
                Sound(((i mod 2)+1)*500);
                Delay(300)
            end;
        NoSound
    END;
    Procedure SelfTest(Len : LongInt);
    Var
        LL : LongInt;
    BEGIN
        CalcFile(LL);
        Writeln(ll);
        IF LL < > Len then
            begin
                Buzzer;         { включаем ревун }
                TextColor(LightRed);
                Writeln;
                Writeln('Внимание! Длина файла изменена.');
                TextColor(Yellow);
                Write(' Эталонное значение..... : ');
                TextColor(LightBlue);
                Writeln(Len:8);
                TextColor(Yellow);
                Write(' Фактическое..... : ');
                TextColor(LightGreen);
                Writeln(LL:8);
                Writeln;
                Halt($F0)
            end
    END;
END.
```

Выходите на связь

Часть 3. Пишем сценарии

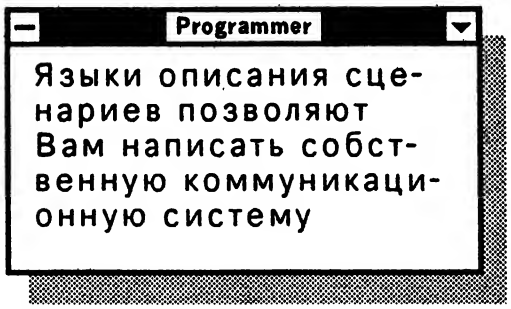
Е.Равич

Одной из привлекательных сторон современных коммуникационных программ является возможность автоматизировать выполнение практически всех этапов связи: от набора телефонного номера до передачи файлов. При желании можно написать собственную достаточно сложную систему коллективного доступа, позволяющую вам и вашим абонентам получить доступ к файлам на различных компьютерах, автоматически принимать и передавать файлы и сообщения в заданное время, общаться глубокой ночью, когда ниже нагрузки на телефонную сеть и чуть выше качество связи. Все это может быть достигнуто, если вы познакомитесь с особенностями языка сценария (scripts), используемого вашей коммуникационной программой, и научитесь описывать на нем все более и более сложные задачи.

В настоящее время известны десятки таких языков, различающихся синтаксисом, набором предлагаемых функций и системных переменных, подходом к созданию и выполнению сценария. Диапазон достаточно широк: от простых языков типа набора нескольких (до 20-30) макроопераций, задающихся мнемонически, до развитых, близких по синтаксису и управляющим конструкциям к привычным языкам программирования высокого уровня с десятками и сотнями встроенных функций. Языки описания сценария (в дальнейшем для краткости просто сценарии) могут снабжаться отдельным компилятором или интерпретатором, встроенным в коммуникационную программу. Как правило, более мощные языки снабжаются компилятором. Это облегчает отладку сценария и увеличивает скорость его выполнения, хотя специальных средств отладки обычно не предусмотрено. Многие коммуникационные программы позволяют протоколировать сеанс связи в специальном режиме обучения (Script Learn), автоматически создающем скелет сценария, который в дальнейшем может быть дополнен и расширен.

Из всего многообразия сценариев мы подробнее рассмотрим SALT (Script Application Language for Telix), используемый в программе Telix, и TMScript программы Telemate. Выбор был сделан по следующим причинам: коммуникационные программы Telix и Telemate достаточно популярны в нашей стране и доступны через многие BBS, синтаксис сценариев аналогичен синтаксису привычных языков программирования, наборы встроенных функций полны и достаточны для решения большинства коммуникационных задач, языки снабжены трансляторами и хорошо документированы. Знание принципов написания

сценариев на этих языках позволяет быстро и безболезненно перейти в случае необходимости к использованию сценариев других коммуникационных программ. Язык SALT достаточно популярен в США среди пользователей BBS, поэтому вы можете использовать большое количество уже созданных и доступных сценариев. Кроме названных языков, имеет смысл обратить внимание на мощные языки CASL семейства программ Crosstalk и Mirror, а также ASPECT программы ProComm Plus 2.0.



SALT : основные возможности и принципы программирования

Версия языка SALT, используемая совместно с программой Telix (версий 3.12, 3.15), представляет собой мощный язык программирования со всеми привычными управляющими конструкциями и синтаксисом, близким к синтаксису языка C. Текст программы набивается в формате ASCII в любом текстовом редакторе. Файл с исходным текстом программы может иметь любое название и расширение, хотя обычно рекомендуется расширение ".SLT". После трансляции файл получает расширение ".SLC". Развитые средства отладки сценария не предоставляются, компилятор CS при обнаружении ошибки останавливается и сообщает номер строки и тип ошибки. Все ошибки должны быть устранены.

Прошедший компиляцию сценарий может быть выполнен следующими способами :
- при запуске Telix может получить имя SLC файла в качестве параметра командной строки и немедленно начать его выполнение;
- в процессе работы программы по команде "Run script";
- сценарий активизируется при выборе конкретного телефона из телефонной книги Telix.

Синтаксис SALT несколько упрощен по сравнению с языком C, в частности, прописные и строчные буквы различаются только в символьных строках. Операторы, названия переменных и функций могут набираться в любом регистре. Комментарии начинаются с "//", могут быть в любом месте программы, причем закрывающей последовательности не предусмотрено. Компилятором игнорируется все, что стоит правее "//" до конца строки.

Допускаются следующие типы данных:
- строчные константы (заключаются в кавычки, могут содержать управляющие символы) и строчные переменные;
- целые переменные (типа int);
- целые константы, которые могут быть шестнадцатиричными;
- метки, используемые в операторах goto.

Строчные переменные могут быть объявлены массивами с размерностью 0-32767. Переменные могут быть глобальными и локальными. SALT поддерживает большинство арифметических и логических операций.

Сценарий может быть разбит на модули, аналогичные функциям языка C. Головной модуль должен иметь имя main. Функции могут принимать аргументы.

В языке имеются следующие управляющие конструкции : if-else (включая вложенность), while, do while, for, return, break, continue, goto.

Сценарий в целом выглядит так:

```
< определение глобальных переменных >
        ...
< определение функции >
< тело функции >
        ...
```

SALT предоставляет более ста встроенных функций высокого уровня, существенно упрощающих проектирование сценария.

Функции подробно описаны в прилагаемом к программе Telix файле документации и снабжены примерами. Кроме функций, программист может использовать свыше сорока системных переменных. В системных переменных отражаются конфигурация Telix, параметры сеанса связи, имена вызываемых внешних программ, имена служебных файлов и каталогов. Доступ к этой информации из сценария дает возможность максимально гибко управлять ходом сеанса, организовывать удаленный доступ с развитой системой парольной защиты и протоколирования, проектировать самонастраивающиеся программы. В целом SALT можно считать одним из наиболее мощных средств написания сценариев среди имеющихся в настоящее время.

В прилагаемом листинге приведен переформатированный отрывок из стандартного сценария HOST.SLT, обеспечивающий работу программы Telix в режиме HOST. Отрывок иллюстрирует логику программирования с использованием SALT при решении достаточно сложных, но часто встречающихся задач, и приводится по следующим соображениям : в нем используется большинство функций и системных переменных SALT, понимание принципов работы коммуникационной программы в режиме HOST бывает важно на практике, а некоторые модули сценария могут использоваться как отдельные программы. В текст сценария были внесены комментарии на русском языке и опущены модули, выполняющие очевидные функции. Полный текст сценария поставляется с программой Telix.

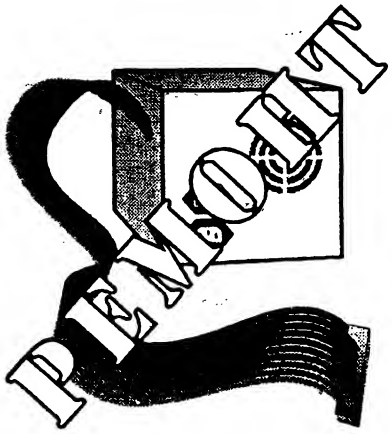
К числу недостатков программы Telix можно отнести отсутствие возможности Learn Mode, то есть автоматического создания скелета сценария в процессе реального сеанса связи. Этот режим уже встроен в меню некоторых коммуникационных программ и позволяет, протоколируя обмен данными и нажатия клавиш, получить исходный текст сценария этого сеанса. Поклонники программы Telix разработали утилиту SLEARN, доступную че-

СРОЧНЫЙ РЕМОНТ БЛОКОВ ПИТАНИЯ

тел. 125 22 12

ВЫ ЗВОНИТЕ – МЫ ПРИЕЗЖАЕМ

- СЖАТЫЕ СРОКИ от 2-х часов до 2-х дней
- На время ремонта ПРЕДОСТАВЛЯЕТСЯ ИСПРАВНЫЙ БЛОК
- ГАРАНТИЯ 3 месяца
- Транспортировка НАШИМ КУРЬЕРОМ



Оплата по безналичному и наличному расчету

Ада-Сервис

117292, Москва, а/я 161, тел. 125 2212

рез некоторые московские BBS, устраняющую этот недостаток. В принципе аналогичные средства может самостоятельно создать любой программист, овладевший языком SALT.

TMScript : еще один мощный язык для создания сценариев

Язык TMScript программы Telemate (версия 3.01) близок по синтаксису и структуре к языку PASCAL, его основные возможности аналогичны SALT, но есть и некоторые важные отличия в технике написания сценария.

Программисту доступны следующие типы данных : целое, строка, булева переменная, дата и время. В TMScript могут использоваться практически все известные управляющие конструкции, в том числе IF-ELSE-ENDIF, SWITCH-CASE-OTHERWISE-ENDSWITCH, WHILE-ENDWHILE, REPEAT-UNTIL, EXIT, RETURN. Язык модульный, но встроенные процедуры TMScript не возвращают значения, а изменяют одну из внутренних переменных программы : CONNECTED, FOUND, SUCCESS, LOGGING, WIDTH и HEIGHT. Смысл значения внутренних переменных зависит от того, какая процедура была перед этим вызвана. Поддерживается операция препроцессора #include.

Процедуры могут быть вложенными друг в друга. Параметры процедуры передаются по значению или по ссылке.

Процедура SET позволяет установить значение для более чем 60 параметров сеанса и конфигурации программы.

Кроме базовых встроенных функций, TMScript позволяет работать с тремя дополнительными наборами сценариев, которые даны в виде исходных текстов и объединены в файлы toolbox1.scr, toolbox2.scr и toolbox3.scr. Доступ к наборам осуществляется, например, с помощью оператора #include. Для уменьшения размеров исходного модуля вы можете исключить неиспользуемые переменные и процедуры.

Набор toolbox1 содержит таблицу цветов и некоторые процедуры для работы с экраном. В toolbox2 даны 3 дополнительные процедуры ввода/вывода и 4 процедуры определения типа символа. Toolbox3 предоставляет 4 процедуры для работы с датой и временем и несколько процедур для работы с телефонной книгой.

В поставляемом вместе с программой Telemate 3.01 файле документации содержится вся необходимая для использования TMScript информация.

Завершая на этом цикл статей, касающихся базовых понятий и возможностей при передаче данных, автор хотел бы поблагодарить читателей за проявленный интерес и пожелать успехов в практической деятельности в области телекоммуникаций. Не претендуя на полноту освещения всех аспектов, основное внимание было уделено базовым понятиям и тем проблемам, с которыми будущий пользователь систем передачи данных сталкивается в первую очередь. В нашей стране использование имеющихся и разработка новых телекоммуникационных систем все еще носит скорее экспериментальный, чем промышленный и, тем более, массовый характер. Только с полномасштабным развитием средств и в первую очередь каналов связи пользователи получат быстрый и дешевый доступ к гигантским информационным ресурсам современного общества, без чего эффективная работа во многих областях становится невозможной.

Все вопросы, замечания, отклики и предложения, направленные в редакцию на имя автора, будут с благодарностью изучены. Автор выражает искреннюю признательность энтузиастам-системным операторам московских узлов сети FIDO, создавшим первую общедоступную некоммерческую сеть в нашей стране, работающую практически в круглосуточном режиме. ■

```
////////////////////////////////////////////////////////////
// HOST.SLT// Copyright (C) 1988,1989,1990,1991 Exis Inc.
////////////////////////////////////////////////////////////
// параметры конфигурации режима HOST

str   pass1[8] = "pass1",                  // Пароль первого уровня
      pass2[8] = "pass2",                  // Пароль второго уровня(Sysop)
      shellpass[8] = "shell", shutpass[8] = "shut",
      host_downloads[64],
      host_uploads[64];

int direct_connect = 0, modem_lock = 0;

str current_caller[31],                    // имя текущего пользователя
      conn300[] = "CONNECT^M", conn1200[] = "CONNECT 1200",
      conn2400[] = "CONNECT 2400", conn9600[] = "CONNECT 9600",
      conn19200[] = "CONNECT 19200";

int finished_caller, local_mode,
      access_level,                        // уровень доступа текущего пользователя
      carrier_counts = 1, already_connected = 0,
      exit_requested = 0, connection_lost = 0,
      kill_user = 0;                       // установка начальных значений флагов

int old_scr_chk_key, old_cisb_auto, old_zmod_auto,
      old_sound;

str old_down_dir[64], old_up_dir[64];

main()
{
      int c;
      clear_scr();

// стирание экрана и попытка считать файл параметров
// конфигурации. В случае ошибок - вывод соответствующего
// сообщения и выход.

if (read_host_config_file() == -1)
{     prints("Unable to read HOST.CNF...");
      prints("Running HCONFIG, the Host Mode configuration
      script.^M^J"); call("HCONFIG");
      if (read_host_config_file() == -1)
      { prints("Still unable to read HOST.CNF. Aborting Host
                  Mode.^M^J"); return -1; }
}
if (!check_directories())
{     prints("Either the upload or download directory as
                  defined in the HOST.CNF file");
      prints("doesn't exist. Either create the missing
                  directory with the DOS 'MKDIR'");
      prints("command, or run the HCONFIG script to redefine
                  what directories to use.");
      prints("Aborting Host Mode.");
      return -1; }
old_scr_chk_key = _scr_chk_key; _scr_chk_key = 0;
old_cisb_auto = _cisb_auto; _cisb_auto = 0;
old_zmod_auto = _zmod_auto; _zmod_auto = 0;
old_sound = _sound_on; _sound_on = 0;
old_down_dir = _down_dir; _down_dir = host_uploads;
old_up_dir = _up_dir; _up_dir = host_downloads;
// сохранение прежних значений системных переменных

usagelog("HOST.LOG");        // открытие файла протокола сеанса
if (direct_connect) carrier_counts = 0;
else carrier_counts = 1;


if (!direct_connect && carrier()) already_connected = 1;
// настройка на текущий сеанс, инициализация модема, переход
// в режим автоответа
while (1)
{     if (!direct_connect && !already_connected)
      {     if (!modem_lock) set_cparams(modem_lock, 0, 8, 1);
            delay(3); prints("Sending Modem Init string...");
            cputs_tr(_mdm_init_str); delay(10);
            prints("Sending Auto-Answer string...");
            cputs_tr(_auto_ans_str); }
      finished_caller = kill_user = 0;
      if (direct_connect) carrier_counts = 0;
      else carrier_counts = 1;
      if (!direct_connect)
      {     prints("^M^JHost Mode: Waiting for call...");
            prints("(Press Esc to exit, or 'L' for local test
                        mode).^M^J");
// ожидание звонка
      do
      {     if (carrier()) { local_mode = 0; break; }
            c = inkey();
            if (c)
            {     if (c == 27) { exit_requested = 1; break; }
                  else if (c == 'l' || c == 'L')
                  {     prints("Local test mode entered");
                        local_mode = 1; carrier_counts = 0; }
      } }

// переход в режим тестирования HOST
      while (toupper(c) != 'L');
      }
      if (!exit_requested)
      {
            prints("Incoming call. Sysop: press Esc to exit, or END
                  to terminate user.");
            do_one_caller();
            if ((connection_lost || kill_user) && carrier_counts &&
            carrier()) hangup();          // при потере связи - выход
      }
      already_connected = 0;
      if (exit_requested)
      {     if (!carrier() && !direct_connect)
            {     prints("Sending Modem Init string...");
                  cputs_tr(_mdm_init_str); }
            _scr_chk_key = old_scr_chk_key;
            _cisb_auto = old_cisb_auto; _zmod_auto = old_zmod_auto;
            _sound_on = old_sound; _down_dir = old_down_dir;
            _up_dir = old_up_dir;
// восстановление значений системных переменных

            prints("^M^JHost mode script finished.");
            usagelog("*CLOSE*");
// сообщение об окончании работы и закрытие файла
// протоколирования сеанса

return 1;
      } } }

////////////////////////////////////////////////////////////
// Модуль HelpOn позволяет получить подсказку по работе в
// режиме HOST, не прерывая сеанс.
////////////////////////////////////////////////////////////

do_one_caller()
{
      str strn[80], fname[64];
      int option, status, c, l, l2, f;

      access_level = 1;
      if (already_connected)
            prints("Already connected, or modem Carrier Detect switch
                  improperly set!");
      else if (carrier_counts)
      {     if (!modem_lock) if (!determine_baud()) ; }
      delay(10); type_file("LOGO.MSG"); flushbuf();
      while (1)
      {     host_send("Please enter your full name: ");
            host_input_strn(current_caller, 30, 1);
            host_send("^M^J");
            if (finished_caller) return;
            if (strlen(current_caller) > = 5) break; }
      access_level = ask_for_pass(3, pass1, pass2);
// вызов модуля проверки пароля. Дается 3 попытки.

if (access_level) ustamp("Logon by ", 1, 0);
else ustamp("Failed logon attempt by ", 1, 0);
ustamp(current_caller, 0, 1);
if (!access_level)
      {     host_send("Goodbye!^M^J");
            if (carrier_counts) { delay(10); hangup(); }
            return; }
      type_file("WELCOME.MSG");
      while (1)
      {
            if (finished_caller) return;
            host_send("^M^JCommand: Help Files Type Upload
                        Download Shell Chat Goodbye ? ");
            host_input_strn(strn, 1, 1);
            option = toupper(subchr(strn, 0)); host_send("^M^J");

            if (option == 'H')
            {     host_send("^M^J^M^JWhich item above do you wish help
```

```
                        on? ");
        host_input_strn(strn, 1, 1);
        Option = toupper(subchr(strn, 0));
        Host_send("^M^J"); HelpOn(Option); Option = 'H'; }

    if (option = = 'F')
    {
        if (access_level = = 2)
        {   host_send("Enter 'filespec' or press Return for
                    *.*,^M^J: ");
            host_input_strn(fname, 64, 1); host_send("^M^J");
            if (just_filename(fname))
            {   strn = host_downloads; strcat(strn, fname); }
            else strn = fname; }
        else
        {   strn = host_downloads; strcat(strn, "*.*"); }

    if (local_mode) show_directory(strn, 0, carrier_counts);
    else show_directory(strn, 1, carrier_counts);
    host_send("^M^J");
    }
    else if (option = = 'T')
    {   host_send("Type what file? ");
        host_input_strn(strn, 64, 1);
        host_send("^M^J");
        if (access_level != 2) fnstrip(strn, 3, fname);
        else fname = strn;
        if (just_filename(fname))
        {   strn = host_downloads; strcat(strn, fname);
            fname = strn; }
        if (!filefind(fname, 0, strn))
        {   host_send("Unable to find "); host_send(fname);
            continue; }
        type_file(fname);
    }
    else if (option = = 'G')
    {   host_send("^M^JGoodbye!^M^J");
        ustamp("User logged off.", 1, 1);
        if (carrier_counts) { delay(10); hangup(); }
        return; }
    else if (option = = 'C')
    {   prints("Sysop: Press Space to chat, any other key not
            to.^M^J"); c = 0; _sound_on = 1;
        for (l = 8; l && !c; --l)
        {   if (carrier_counts && !carrier())
            {   prints("^M^JConnection has been lost, call
                        terminated.^M^J");
                connection_lost = 1; finished_caller = 1;
                break; }
            cputc('^G'); tone(523, 20);
            tone(659, 20); tone(523, 20); tone(659, 20);
            tone(523, 20); tone(659, 20);
            for (l2 = 30; l2 && (c = inkey()) = = 0; --l2)
            delay(1); }

        _sound_on = 0;
        if (finished_caller) continue;
        if (c != '' || !c)
        {   host_send("Sorry, the Sysop is unavailable^M^J");
            continue; }
        host_send("The sysop is here!^M^J"); chatmode(1);
        }
        else if (option = = 'U')
        {   option = host_get_prot();
            if (!option) continue;
            status = 1;
            if (option = = 'T' || option = = 'M' || option = = 'S' ||
                option = = 'Y' || option = = 'Z' || option = = 'E')
            {   send_transfer_msg(); status = receive(option, ""); }
            else
            {   host_send("Upload what file? ");
                host_input_strn(strn, 48, 1); host_send("^M^J");
                if (!strn) continue;
                if (access_level != 2) fnstrip(strn, 3, fname);
                else fname = strn;
                if (just_filename(fname))
                {   strn = host_uploads; strcat(strn, fname);
                    fname = strn; }
                if (filefind(fname, 23, strn))
                host_send("File already exists!^M^J");
                else {      send_transfer_msg();
                            status = receive(option, fname); }
                }
                if (status = = -2)    // потеря несущей
                connection_lost = finished_caller = 1;
                else if (status = = -1)
                host_send("^GOne or more files not received!^M^J");
            }
            else if (option = = 'D')
            {   option = host_get_prot();
                if (!option) continue;
                host_send("Download what file(s)? ");
                host_input_strn(strn, 48, 1); host_send("^M^J");
                if (!strn) continue;
                if (access_level != 2) fnstrip(strn, 3, fname);
                else fname = strn;
                if (just_filename(fname))
                {   strn = host_downloads; strcat(strn, fname);
                    fname = strn; }
                if (!filefind(fname, 0, strn))
                {   host_send("Unable to find any matching
                            file(s)!^M^J"); continue; }
                status = 1; send_transfer_msg();
                status = send(option, fname);

                if (status = = -2)
                connection_lost = finished_caller = 1;
                else if (status = = -1)
                host_send("^GOne or more files not received!^M^J");
            }
            else if (option = = 'S')          // режим удаленного доступа
            {
                if (ask_for_pass(3, shellpass, shellpass) != 0)
                {
                    host_send("Type EXIT and then press Enter to come
                            back.^M^J");
                    if (get_baud() = = 300) delay(10);
                    if (local_mode) dos("", 0);
                    else if (filefind("RSHELL.BAT", 0, strn))
                        dos("RSHELL.BAT", 0);
                    else
                    {   f = fopen("HOSTTEMP.BAT", "w");
                        if (f)
                        {   if (get_port() != 1 && get_port() != 2)
                            { host_send("Remote Shell not supported
                            on this comm port due to DOS limits!^M^J");
                            continue; }
                        strn = "COMx"; setchr(strn, 3, get_port() + '0');
                        fputs("CTTY ", f); fputs(strn, f);
                        fputs("^M^JCOMMAND^M^J", f);
                        fputs("CTTY CON^M^J", f); fputs("EXIT^M^J", f);
                        fclose(f);
                        if (!local_mode) dos("HOSTTEMP.BAT", 0);
                        }
                        else
                        host_send("Can't open temporary batch file!^M^J");
    } } }
    else if (option = = '^Z')          // выход из режима HOST
    {   host_send("Shut down Host mode. ");
        if (!ask_for_pass(3, shutpass, shutpass)) continue;
        host_send("Goodbye!^M^J");
        if (carrier_counts) hangup();
        ustamp("User shut down Host Mode.", 1, 1);
        finished_caller = 1; exit_requested = 1;
    } } }

////////////////////////////////////////////////////////////
 // запрос на использование одного из имеющихся протоколов
host_get_prot()
////////////////////////////////////////////////////////////
send_transfer_msg()
{   host_send("Ready to transfer file(s)... Press Ctrl-X at
            least twice to abort^M^J"); }
////////////////////////////////////////////////////////////

determine_baud()
// Модуль определяет скорость вызывающего модема и выводит
// соответствующее сообщение
{
    int t3, t12, t24, t96, t192, tmark, stat, new_baud = 0;

    printsc("Determining baud... "); track_free(0);
    t3 = track(conn300, 0); t12 = track(conn1200, 0);
    t24 = track(conn2400, 0); t96 = track(conn9600, 0);
```

```
    t192 = track(conn19200, 0);
    tmark = timer_start(30);          // трехсекундное ожидание...
    while (!time_up(tmark))
    {   if (!carrier())
        {   track_free(0); return 0; }
        if (cinp_cnt()) track_addchr(cgetc());
        stat = track_hit(0); if (stat = = 0) continue;
        if (stat = = t3) new_baud = 300;
        else if (stat = = t24) new_baud = 2400;
        else if (stat = = t96) new_baud = 9600;
        else if (stat = = t192)new_baud = 19200;
        else new_baud = 1200;
        break;   }
    if (!new_baud)
    {   prints("Failed!"); track_free(0); return 0; }
    printn(new_baud); prints(""); set_cparams(new_baud, get_parity(),
    get_datab(), get_stopb()); track_free(0);
    return 1;
}

type_file(str fname)
{
    int f;
    str buf[100];
    int ichar, lines_sent = 0;

    f = fopen(fname, "r"); if (!f) return -1;
    host_send("^M^J");
    while (1)
    {   if (carrier_counts)
            if (!carrier())
            {   connection_lost = 1; finished_caller = 1;
                fclose(f); return 0; }
            if (fgets(buf, 80, f) = = -1) { fclose(f); return 1;}
        host_send(buf); host_send("^M^J"); + +lines_sent;
        if (lines_sent > = 22)
        {   lines_sent = 0; host_send("[More]"); host_input(1);
            if (finished_caller) { fclose(f); return 0; }
        host_send("^H ^H^H ^H^H ^H^H ^H^H ^H^H ^H^H ^H^H ^H"); }
    while (cinp_cnt())
    {   ichar = cgetc(); if (ichar = = '^C' || ichar = = '^K')
        {   host_send("^M^J"); fclose(f); return 1; }
} } }

host_send(str outstr)
{
    printsc(outstr);
    if (!local_mode) cputs(outstr);
}

host_send_c(int chr)
{
    printc(chr);
    if (!local_mode) cputc(chr);
}

host_input_strn(str buf, int maximum, int echoable)
{
    int i = 0, key;

    while (1)
    {
        key = host_input(echoable);
        if (!key)
        {   setchr(buf, 0, 0); return 0; }
        if (key = = '^M') break;
        if (key = = 127 || key = = 8)
        {   if (i) { —i; host_send_c(key); } continue; }
        if (i < maximum) { setchr(buf, i, key); i = i + 1; }
        else i = i + 1;
    }
    if (i > maximum) i = maximum; setchr(buf, i, '^0');
    if (subchr(buf, 0)) return 1; else return 0;
}

host_input(int echoable)
{
    int c, t;

    t = timer_start(2400);
    while (1)
    {
        if (time_up(t) && !direct_connect)
        {   host_send("^M^J^M^JInactivity period too long.
                       Connection terminated!^M^J");
            if (carrier_counts) hangup(); finished_caller = 1;
            kill_user = 1; return 0; }
        if (carrier_counts)
            if (!carrier())
            {   prints("^M^JConnection has been lost, call
                            terminated.^M^J");
            connection_lost = 1; finished_caller = 1; return 0; }
        if ((c = inkey()) ! = 0)
        {   if (c = = 27)
            {   finished_caller = 1; exit_requested = 1; return 0;}
            else if (c = = 0x4f00)
            {   prints("^M^JUser terminated!");
                ustamp("User terminated!", 1, 1);
                if (carrier_counts) hangup(); finished_caller = 1;
                kill_user = 1; return 0; }
        else if (c < = 255)
        {   if (c ! = 8 && c ! = 127)
            if (!echoable) host_send_c('*');
            else host_send_c(c); return c; } }
        if (!local_mode) if (cinp_cnt())
        {   c = cgetc();
            if (c ! = 8 && c ! = 127)
            if (!echoable) host_send_c('*');
            else host_send_c(c);
            return c; }
    } }

ask_for_pass(int maxtries, str pass1, str pass2)
// Модуль запрашивает пароль пользователя. После трех ошибок
// выводит сообщение о разрыве связи.
{
    int i;
    str strn[8];

    for (i = 0; i < maxtries; + +i)
    {   if (i) host_send("Wrong! Try again.^M^J");
        host_send("Password: "); host_input_strn(strn, 8, 0);
        host_send("^M^J");
        if (finished_caller) return 0;
        if (!strcmpi(strn, pass1)) return 1;
        if (!strcmpi(strn, pass2)) return 2; }
    host_send("No more chances. Access denied.^M^J");
    return 0;
}

////////////////////////////////////////////////////////////
// считывание файла конфигурации аналогично сценарию
// HCONFIG.SLT // read_host_config_file()
////////////////////////////////////////////////////////////

// проверка наличия каталогов - параметров конфигурации
check_directories()
{
    str s[80];
    int i, a;

    s = host_uploads; i = strlen(s);
    if (i > 0)
        if (subchr(s, i - 1) = = '\' || subchr(s, i - 1) = = '/')
            setchr(s, i - 1, 0);
    if (s && !(strlen(s) = = 2 && subchr(s, 1) = = ':'))
    {   a = fileattr(s);
        if (a = = -1 || !(a & 16)) return 0; }
    s = host_downloads; i = strlen(s);
    if (i > 0)
        if (subchr(s, i - 1) = = '\' || subchr(s, i - 1) = = '/')
            setchr(s, i - 1, 0);
    if (s && !(strlen(s) = = 2 && subchr(s, 1) = = ':'))
    {   a = fileattr(s);
        if (a = = -1 || !(a & 16)) return 0; }
    return 1;
}

////////////////////////////////////////////////////////////
// проверка правильности имени файла
// just_filename(str filespec)
////////////////////////////////////////////////////////////
```

■

UNICOM HAYES MODEM 2400bps*

НАТЕКС

- **Современный протокол CCITT V.22bis, скорость 2400 бит/с — это совместимость с 99% мирового парка модемов.**
- **Программная эмуляция MNP2—7 многократно повысит реальную скорость передачи и гарантирует отсутствие ошибок.**
- **В основе модема — самый современный процессор фирмы Rockwell International Corporation, USA. Производство плат осуществляется на новейшей импортной технологической линии с использованием строгих методов контроля.**

"НАТЕКС" обеспечивает:

— установку модема и обучение персонала

— постоянные консультации специалистов по аппаратному и программному обеспечению

— стабильность поставки

— гарантийное обслуживание— 2 года

— постгарантийный сервис

Модем НМ—2400 предназначен для работы на ПЭВМ семейства IBM PC XT/AT/AT386.
Модем представляет собой плату, которая встраивается в любой свободный слог расширения компьютера и подключается к телефонной линии параллельно телефонному аппарату.

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

- **Расширенный AT набор команд**
- **Протоколы CCITT V.22bis, V.22, V.21, Bell 103, 212A**
- **Скорость 2400, 1200, 0—300 бит/с**
- **Автоответ, автонабор, распознавание сигналов линии**
- **Автоматический выбор скорости и распознавание протоколов**
- **Тональный и импульсный набор**
- **Работа на телефонной линии или витой паре**
- **Встроенный динамик с регулировкой громкости**
- **Запоминание телефонного* номера до 33 знаков**
- **Шнур с телефонными вилкой и розеткой отечественного стандарта**

В комплект поставки входит руководство на русском языке с описанием AT команд
*** HAYES MICROCOMPUTER PRODUCT, INC, USA**

Если Вас заинтересовала наша информация, звоните по телефону: 324—40—00. Почтовый адрес: 103031 Москва, а/я 839, НТЦ "НАТЕКС"

НАТЕКС вырывается вперед!

Вы уже обратили внимание на рекламу, помещенную на этой странице?

Если нет, рекомендуем вам вчитаться повнимательнее. НАТЕКС не перепродает, а производит модемы, функционирующие в соответствии со стандартами V.22 bis и V.22!

Если вы еще не прочувствовали, добавим пару фраз.

Модемов в России производится много и, надо отметить, очень разных. Это и нашумевший "Лександ", и "Комп", и "Мастак" и т.д. и т.п. Но все эти модели объединяет то, что каждая из них способна "разговаривать" только с модемом той же модели. А называется это очень красиво : "...адаптация к советским линиям телефонной связи".

И все казалось бы хорошо и правильно, да вот незадача: не хотят отечественные любители и фанаты "адаптироваться", хотят они, космополиты этакие, общаться со всем миром, да еще по ихним, по буржуйским стандартам. Ну что же, сколько пива, столько песен... И появляются на нашем зарождающемся рынке импортные модемы, и начинаешь понемногу сравнивать рубль с японской йеной или итальянской лирой, и начинаешь скрежетать зубами от черной зависти к счастливым обладателям буржуйских модемов. И клянешь всеми нехорошими словами, которыми научили еще при изучении курса истории КПСС и марксистско-ленинской философии отечественных разработчиков, которые, уповая на иллюзорную мечту о монополизации рынка телекоммуникаций, делают ни с чем не совместимые модемы.

И вот, наконец, забрезжила надежда. Научно-технический центр НАТЕКС освоил производство "настоящих" модемов, обеспечивающих обмен со скоростью до 2 400 Бод, поддерживающих расширенный набор AT команд, т.е. долгожданный Hayes-модем.

Но стоит ли доверять фирме, о которой вы раньше не слышали? Решайте сами, сейчас самое время познакомиться!

Научно-технический центр НАТЕКС был создан в январе 1991 года как общество с ограниченной ответственностью. Уставной фонд общества составляет 75 тыс. рублей.

Предприятие создано как развитие хозрасчетного научно-производственного отдела совместного предприятия ИНПРИБОР, который действовал с начала 1990г.

Основными направлениями деятельности предприятия являются:

– разработка и производство устройств расширения для ПЭВМ, преимущественно для IBM PC, таких как различные цифровые и аналоговые интерфейсы, контроллеры, адаптеры;

– разработка и производство телекоммуникационного оборудования – аппаратных средств, программного обеспечения, создание и эксплуатация компьютерных сетей;

– разработка и внедрение в производство систем технической диагностики для нефтегазовой и химической промышленности;

– разработка и опытное производство различных уникальных приборов и оборудования, в том числе для медицинского применения.

НТЦ НАТЕКС обладает на настоящий момент налаженными производственными связями с рядом крупных промышленных предприятий в Москве, Московской обл., С.-Петербурге, что позволяет центру обеспечить бесперебойное производство продукции в условиях нестабильной экономической ситуации в стране.

Центром решена проблема снабжения собственного производства необходимыми комплектующими изделиями. Технология производства печатных плат и монтажа, используемая центром, основана на импортном оборудовании и материалах, что обеспечивает качество, соответствующее мировому уровню.

Кроме того, если вы приобрели модем у фирмы НАТЕКС, вам обеспечено двухлетнее гарантийное обслуживание и постгарантийный сервис. Автор рискнет открыть небольшой секрет разработчиков радиоэлектронной аппаратуры – двухлетний срок гарантии на изделие обычно означает, что средний срок наработки на отказ по результатам тестирования опытной партии составляет не меньше 3,5–4 лет.

Да и само по себе предложение гарантийного обслуживания в значительной мере облегчает жизнь вам, как потребителю. А если добавить к этому консультации специалистов фирмы, то остается только еще раз повторить:

НАТЕКС вырывается вперед!"

Новое лицо редактора ChiWriter – версия 4 !

А. Дайховский

Несколько лет назад, когда я еще редко работал на компьютере, моим любимым текстовым редактором был Лексикон. Для заявлений, отчетов и служебных записок он вполне годился, и я горя с ним не знал. Но однажды мне понадобилось набрать текст с химическими формулами и математическими знаками. Изобразить подстрочные и надстрочные символы Лексикон отказался наотрез, и мне пришлось обратиться за помощью к более искушенным пользователям, которые и порекомендовали ChiWriter. "Да тут же все на английском !" - была моя первая реакция (я учил французский). Не было ничего: ни документации, ни мудрого помощника за плечом. Но выбирать не приходилось, и я начал работать. Не прошло и часа, как были готовы первые формулы, а на следующий день я уже вполне свободно обращался с новым редактором. С тех пор с Лексиконом я больше не работал.Зачем? Ведь ChiWriter обладал всеми функциями стандартного текстового редактора: копированием строк и прямоугольных областей текста, динамической нумерацией страниц, наличием сносок и заголовков с любым количеством строк, над- и подуровнями, автоматической нумерацией сносок, созданием таблиц, вставкой рисунков (на этапе вывода на принтер), выравниванием границ текста с полуавтоматическим переносом части слова, поиском и заменой фрагментов текста, проверкой орфографии английских слов и т.д. К тому же он обладал таким ценным качеством, как работа в режиме "WYSIWYG" (что видишь на экране, то будет и на принтере). А те 25 английских слов, что встречались в меню, я выучил до конца недели.

Это была третья версия ChiWriter. И вот спустя пару лет, когда я получил в свои руки ChiWriter v.4.02, меня первым делом удивил размер пакета. Согласитесь, 14 дискет по 360 килобайтов - это многообещающе. Первые 8 дискет занимает сам редактор (с набором шрифтов для большого количества матричных и лазерных принтеров). Еще 3 дискеты - это химические, электрические и русские шрифты. И плюс ко всему программы для автоматического составления оглавления и указателя (glossary, index), конвертер текстов из ChiWriter v.4.02 в популярный научный редактор TeX и конвертер из WordPerfect и WordStar в ChiWriter v.3.xx-4.02 (8-битовые ASCII тексты ChiWriter импортирует-экспортирует автоматически, в том числе и русские).

Скажу сразу, что многие команды для удобства совпадают с 3-й версией, меню по-прежнему находится внизу, а строка статуса - вверху. Но на этом основное сходство и заканчивается. Теперь можно забыть о старой версии, как о совсем другом редакторе. Практически все ее бывшие минусы превратились в плюсы. Дополнительные меню появляются в середине экрана, а передвижение курсора по тексту отображается на экране в сантиметрах, дюймах или строках/колонках. Десять окон позволяют работать параллельно с десятью документами, а объем каждого документа неограничен. Текст, формулы и таблицы форматируются автоматически, при этом формулы могут быть в любом месте документа. Введен новый специальный режим GRID для рисования прямоугольников, диаграмм и любых других квазиграфических рисунков. Возможность ввода и печати одного символа поверх другого позволяет создавать весьма сложные и полезные фигуры, например сплошное пересечение линий. Для удобства пользователей применяется автоматическая нумерация не только страниц, но и рисунков, ссылок, формул, таблиц, глав, параграфов и т.п. А при добавлении, например, новой таблицы в середину текста, соответственно меняется нумерация самих таблиц и ссылок на них во всем документе. Добавлен режим создания таблиц на несколько страниц. Появились две новые прекрасные команды - UNDO и REDO. Это значит, что теперь можно вообще не бояться сделать какие-либо ошибки - в любой момент вы можете отменить по шагам свои последние сто(!) команд. Я даже специально "играл собачий вальс" на клавиатуре, а потом полностью восстанавливал первоначальный текст. Поверьте, это доставило удовольствие.

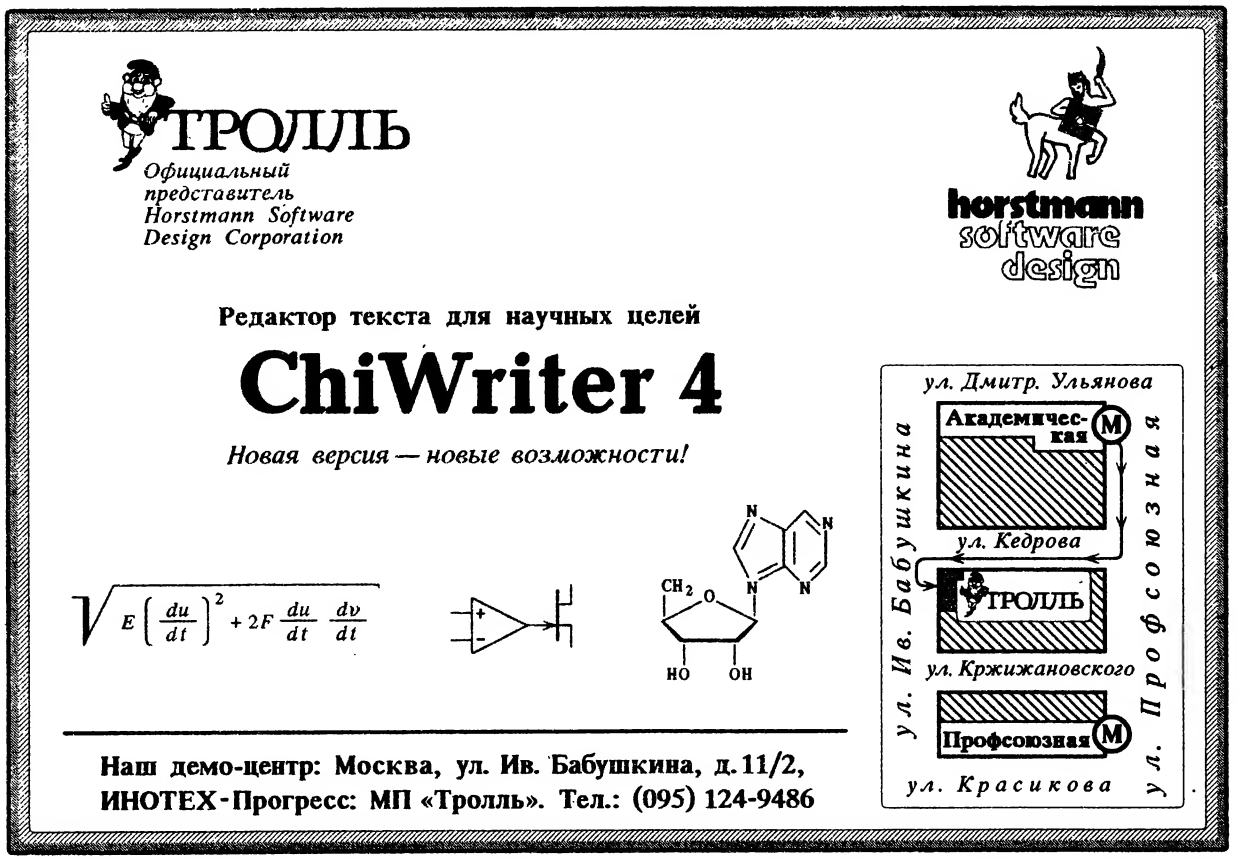
Редактирование любых формул и таблиц теперь также доставит удовольствие. Это происходит так: из набора заготовок формул выбираете, допустим, квадратный корень и вписываете под него все, что вам нужно. Заготовка будет сама автоматически раздвигаться вправо, влево, вверх или вниз, чтобы вместить ваши знаки.Чрезвычайно просто теперь создаются таблицы. Для этого надо лишь указать количество строк и колонок и больше не думать о

разделительных рамках. Они будут меняться соответствующим образом сами при вводе информации. При этом можно задавать формат каждой колонки в отдельности: с выравниванием по любому краю, по центру и знаку, например по десятичной точке. Все вышеперечисленное относится и к матрицам любого размера. Остается добавить только, что при печати таблиц пропорциональными шрифтами вы увидите, что напечатали действительно красивую таблицу, а не "шведскую стенку" после удара каратиста.

ChiWriter v.4.02 позволяет создавать и использовать макросы, определяемые последовательностями нажатия клавиш. В дистрибутив входят наборы макросов для изображения сложных математических символов, структурных химических формул (в том числе в изометрической проекции) и электрических принципиальных схем. Выбираете из набора бензольное кольцо и в нужном месте вписываете радикал. Формула готова, причем быстрее, чем если бы рисовали от руки. Прекрасный подарок для научного работника!

Теперь несколько слов о шрифтах. В новой версии ChiWriter для кодировки используются все 255 символов, а это значит, что в одном шрифте можно объединить два - русский и английский. Таким образом, работать с текстом в ChiWriter v.4.02 можно, имея одновременно сорок обычных шрифтов, а не двадцать, как раньше. Легко также сделать русскоязычное меню, хотя мне вполне хватило учебника на русском языке. В дистрибутив входит и усовершенствованная версия Font Designer (редактора шрифтов для ChiWriter). С его помощью вы можете легко редактировать старые и создавать любые новые шрифты: от старославянского до готического. Те, кто работал с ним раньше, заметят, что делать это стало намного удобнее: менять высоту, ширину, базовую линию, тип шрифта, скалировать его к определенному размеру (с высоким качеством), копировать отдельные буквы и т.д. Добавлю, что FontDesigner импортирует лазерные шрифты в форматах ChiWriter и HP LaserJet (portrait), а экспортирует в форматы ChiWriter, DeskJet, Postscript и HP LaserJet (portrait и landscape)!

Насколько мне известно, в 1992 году фирма-производитель Horstmann Software Design Corp. выпускает новую версию ChiWriter - 4.1. Что же будет нового на этот раз? Это – импорт графики в форматах TIFF и PCX прямо в текст на экран, ее редактирование непосредственно в редакторе (надписи поверх графического образа, изменение размеров или удаление части картинки) плюс работа с "мышью". Кроме того, в ChiWriter v.4 будут встроены корректор опечаток русского текста и блок автоматического переноса русских слов. Все это сделает работу пользователя еще более приятной и удобной. Поработав с ChiWriter v. 4, Вы возможно согласитесь, что при своей простоте в управлении, обучении и наличии документации на русском языке, ChiWriter становится довольно серьезным конкурентом для известных издательских систем. ■



Начало на стр. 8

После этого, заявляя, что на первый вопрос читатель ответит самостоятельно (вероятно так читают лекции во Флоридском университете), автор начинает отвечать, хотя и весьма убого, на второй, а затем претендует на подробное освещение третьего.

Не в обиду автору будет сказано, но "теоретическая часть" выглядит весьма скромно. Даже в "Компьютерной вирусологии" эти вопросы освещены гораздо подробнее.

В то же время следует отметить глубокие познания автора в организации и функционировании операционной системы MS-DOS, которыми он щедро делится с читателями. Чего стоит, например, заявление:

"Поскольку в .COM-файле, если не принимать специальных мер, адрес входа всегда равен 100h, то для передачи управления по этому адресу вирус может использовать длинный переход и не рассчитывать относительное смещение адреса, учитывающее длину заражаемой программы" (стр.6).

Блестяще! Но какие меры надо предпринять, чтобы адрес входа COM файла начал отличаться от 100h, автор не сообщает. Я кстати также не могу помочь непонятливому читателю, потому что операционная система при загрузке COM файла ВСЕГДА (несмотря на специальные меры П.Л.Хижняка) передает управление по адресу CS:100h! А что касается длинного перехода, то я хочу порадовать Петра Львовича. Не нужен ни длинный переход, ни вычисление каких -то смещений. Начиная с самых первых вирусов (DOS-62, он же Vienna), эта операция выполняется следующим образом:

```
mov   di,100h
push  di
ret
```

И все! И незачем морочить читателю голову!

Ну да ладно. Сорвался, позвольте замереть перед вами в глубоком пардоне. Это все, как говорится, цветочки. Я согласен, что у каждого может быть свой взгляд на разработку программного обеспечения и навязывать какое-то решение просто неприлично. Но идем дальше.

"Не предполагая у читателя досконального знания языка ассемблера, прерываний и функций MS-DOS и BIOS, мы будем давать по возможности полные комментарии к каждой структурной части программы-вируса и антивируса" (стр. 10).

И на том спасибо! Однако, перелистав страницы, обнаруживаешь, что все комментарии в листингах программы написаны на английском. Конечно читателю приятно полагать, что его считают не глупее студента Флоридского университета, но на мой взгляд, книжка для русского человека должна быть написана по-русски.

Ну хватит цепляться! - начинает негодовать читатель, - все равно любой пользователь IBM должен хоть немного понимать по-английски. Вот пусть и подучится.

Но на это возражение у меня тоже припасен ответ (а иначе я бы и не стал давать слово сердитому читателю). Чему хорошему может научиться читатель при изобилии грамматических ошибок в английском тексте? Примеры? Пожалуйста.

Во-первых, известное всем программистам слово Hello нигде не пишется как Hallo (хотя во Флориде я не бывал). Во-вторых, при разработке антивируса автор предлагает использовать для включения фага ключ /q - от слова qure, которое якобы означает "лечить" (стр. 47). На самом же деле, это слово пишется : cure, что и подтверждают многочисленные примеры комментариев в листинге программы.

Как и обещано было в аннотации к брошюре (ну не могу я называть этот опус книгой, не могу!), действительно приведены исходные тексты вируса, поражающего COM файлы, и детектора-фага для него. Я не буду детально анализировать коды приведенных в книге ассемблерных программ, поскольку это займет слишком много места и поможет только начинающим писателям вирусов. Хочу только отметить, что если ты, читатель, начнешь писать вирусы по мотивам произведения П.Л.Хижняка, то Н.Н.Безруков может спать спокойно - ты никогда не взорвешь реактор и не запустишь баллистическую ракету. Ты все время будешь бороться с несуществующими трудностями и писать настолько замысловатый код, что не сможешь разобраться в нем и через две недели! Я понимаю, что обвинения выглядят достаточно голословными, но не приводить же на страницах журанала вирус, выполняющий все те же функции, но короче в два раза!

А теперь коротко подведем итоги. Прежде всего хочу сослаться на послесловие к брошюре.

"...Если не пренебрегать правилами вирусной безопасности и не перенасыщать безо всякой системы ваш винчестер играми и другими программами с сомнительным происхождением, можно вполне выйти из этой борьбы, если не победителем, то уж во всяком случае без существенных потерь. Например, у автора этой книги, который чуть ли не каждую неделю борется с вирусами, проникающими на компьютеры его друзей, за два последних года в результате применения системы мер безопасности не было замечено (тьфу-тьфу!) ни одного вируса. А если вирус все же появится, надо быть во всеоружии. Именно на то, чтобы дать читателю такое оружие или хотя бы познакомить с тем, как оно куется, и нацелена данная книга. Удачи!"

Я готов согласиться только с "тьфу-тьфу!" и с последним словом. Да, читатель, тебе действительно потребуется невероятное везение, чтобы, прочитав эту брошюру, "быть во всеоружии"!

Что касается системы мер безопасности, которые, как читатель наверное уже догадался, были разработаны "именно там" (см.выше), то они представляют собой краткий (ну очень краткий) конспект рекомендаций Н.Н.Безрукова и уместились на половине страницы. Господа спонсоры! Свозите меня почитать лекции в Штаты, я напишу две страницы подобных сентенций!

Итак, что же мы имеем за 10 рублей? Из трех вопросов (по три рубля тридцать три копейки каждый) мы не получили вразумительного ответа ни на один! Первый вопрос (откуда взялись вирусы?) автор просто "зажилил". Второй (что это такое?) так и остался вопросом. Хотя, как говорится, умный догадается, а дураку и не нужно. Третий вопрос (как с этим бороться?) естественно тоже остался без ответа.

А поэтому ответим на главный вопрос, который волнует читателя, покупать ли книжку? Если вы хотите научится бороться с вирусами, то наверное нет. Ничего интересного вы здесь не найдете.

Если вы хотите научиться писать вирусы, то тоже не стоит - по этой книге хорошо врага запутывать, а не чему-то учиться.

А вот если вы хотите написать на эту книгу злую рецензию, как это сделал я - не жалейте червонца! Здесь столько ляпов, что хватит надолго! Удачи Вам, господа!

В.Водолазкий ■

Литература

1. **П.Л.Хижняк. Пишем вирус и антивирус. П/ред. И.М.Овсянниковой. - М: ИНТО, 1991, - 90 с.**
2. **Н.Н.Безруков. Компьютерная вирусология. Справочное руководство. К: УРЕ, 1991.**

" Ты записался в убийцы ДЖАФАРА ??? "

Полезные советы любителям игры " Prince of Persia "

А. Лукашук

Рано или поздно первые уровни Вы научитесь проходить легко и они откровенно Вам наскучат, в то время как следующих уровней (всего 12) Вы просто не видели.

Фирма-разработчик	**: Broderbund Software**
Графика	**: Hercules/CGA/Tandy/ EGA/VGA**
Звук	**: Adlib/Covox/ Game Blaster/ IBM Music Feature/ Sound**
Управление	**: Keyboard/Joystick**

"Prince of Persia" - одна из увлекательных и динамичных игр, правдами и неправдами дошедших до остатков нашей великой державы.

Идея игры достаточно проста, что впрочем является одним из преимуществ ее перед многими другими играми подобного класса. "Нехороший" султан Джафар захватил прекрасную девушку - возлюбленную нашего героя Принца Персии. Вы в роли самоотверженного молодого человека должны за 60 минут спасти Принцессу и убить Султана. На Вас будут нападать слуги Джафара, незаметные провалы и ловушки постоянно будут сопровождать Вас на пути к победе, но удача капризна. А времени будет катастрофически не хватать.

Рано или поздно первые уровни Вы научитесь проходить легко и они откровенно Вам наскучат, в то время как следующих уровней (всего 12) Вы просто не видели. Необходимость увеличения жизненного и боевого потенциала Принца и стала подоплекой рождения настоящей заметки.

Наши доблестные "крэкеры" вычислили "Hot Key", который по всей видимости использовался разработчиками для отладки программ, врочем не исключено, что это специальный пароль, высылаемый за отдельную плату.

Итак, искомое " секретное " слово :

!!! MEGAHIT !!!

То есть теперь вместо традиционного запуска " prince.exe ", Вы в командной строке должны набрать:

```
prince megahit
```

Таким образом, запуская просто "prince.exe", Вы имели следующий набор средств.

Команды управления:

Up	– прыжок вперед / прыжок вверх
Down	– присесть / спуститься вниз
Left	– движение влево
Right	– движение вправо
Left/Up	– прыжок влево / длинный прыжок
Right/Up	– прыжок вправо / длинный прыжок
Right-shift/Joystick button	– поднять что-либо / удар шпагой
Ctrl-A	– перезапустить текущий уровень
Ctrl-R	– возврат на первый уровень
Ctrl-S	– звук вкл / выкл
Ctrl-V	– версия
Ctrl-Q	– выход в DOS
ESC	– пауза
SpaceBar	– индикация времени игры

После запуска игры с заветным " megahit " Вы(помимо вышеперечисленных команд) будете располагать следующими крайне полезными органами управления ходом игры.

Команды управления:

Ctrl-{	– пауза
Ctrl-K	– управление от клавиатуры
Ctrl-J	– управление от джойстика
Shift-W	– принц может прыгать, не разбиваясь
Shift-T	– добавляет жизнь (макс. 10)
Shift-I	– инверсия изображения
Shift-+	– добавляет время игры (макс.16 час 40 мин)
Shift-B	– выделение динамически изменяющихся объектов (отсечение фона)
K	– убивает противника (кроме скелетов)
U	– обзор экрана сверху принца
N	– обзор экрана снизу принца
H	– обзор экрана справа от принца
J	– обзор экрана слева от принца

Удачи Вам, господа !!! ■

старт на запад

Наш бензин для Ваших идей

Фирма Spirit, занимающаяся экспортом know–how, приглашает разработчиков оригинальных программных продуктов к сотрудничеству.
Рассматриваются любые завершенные проекты, основанные на собственной технологии.

Дух Свободных Идей
of Free Ideas

Специалисты фирмы Spirit

проведут	*экспертизу Вашего проекта*	**найдут**	*западного инвестора, заинтересованного в Ваших разработках*
подготовят	*рекламу Вашей программы на английском языке*	**помогут**	*Вам перевести интерфейс и документацию программы на английский или немецкий языки*
организуют	*рекламную кампанию в Германии и США*		

Для рассмотрения Вашего проекта высылайте подробное описание, демонстрационную версию и квитанцию о денежном переводе (435 рублей для частных лиц и 2450 рублей для организаций) по адресу: **Москва, 117454, а/я 63. С пометкой "Трансфер".**

Счет: р/с 0467072 в КБ "ИнвестБизнесБанк" корр. счет 1161572 МФО 201791 в ГУ ЦБ РФ по г. Москве

Тел. **(095) 457– 9216, 138– 8781**
Факс **(095) 126—3156**

Software Products Integrated Research Intelligence Transfer

Clipper 5

ВЫБОР ПРОФЕССИОНАЛОВ

Став зарегистрированным пользователем, Вы сможете:

- **обладать полностью русифицированным продуктом и документацией (за рубли);**
- **легально разрабатывать и распространять собственные коммерческие продукты на Clipper;**
- **получать бесперебойные консультации по "горячей" телефонной линии (095) 289– 43– 00.**

Легальная копия– входной билет в мировое сообщество Clipper

За информацией обращайтесь в СП "Магнит".
Наш адрес: 127018, Москва, 2– я Ямская, 15.
Телефон: (095) 289– 44– 77 Факс: (095) 289– 34– 69

Nantucket®

Зак. 969 Тир. 25.000 Ф-ка «Картолитография»